Escuela Universitaria de Ingeniería
Técnica de Telecomunicación
Universidad Politécnica de Madrid

Proyecto Final de Carrera

Aplicación móvil de localización de ocio para Android.

Autor: Eduardo Campos de Diago
Legajo: 54483
DNI: 51423674-K
Tutor: Dra. Silvia Gómez

2013

Dedicado a
mis padres y hermana por apoyarme en todo momento

II

Agradecimientos

A mis amigos Tono, Mardo, Beto y Rober porque sus risas son mis alegrías.

A mis colegas de la universidad: Tiko, Kike, Paula, David, desArmando... Porque nuestras cabezas se han llenado juntas de nuevas paranoias.

A mi tutora Silvia Gómez y a Leticia Gómez, que desde las sombras me ha echado un cable.

MUCHAS GRACIAS!

Resumen

¿No sabes dónde comer un plato típico cuando estás de vacaciones? ¿Quieres ir con amigos a comer a un sitio distinto? ¿Quieres disfrutar de esa comida que tanto te gusta y no sabes dónde hacerlo? Con afán de responder a estas preguntas y gracias a las capacidades que nos brindan las nuevas tecnologías de dispositivos móviles, surge la aplicación que se presenta en este proyecto fin de carrera. Se trata de una aplicación para dispositivos móviles con sistema operativo Android que nos brindará la opción de encontrar restaurantes en nuestro entorno que nos ofrezcan esa comida que queremos. Además, a modo de red social, incluye la opción de poder puntuar los platos degustados en los restaurantes e insertar restaurantes nuevos, lo que hace que la aplicación tenga una mayor versatilidad.

En este documento se podrán encontrar los diagramas UML que modelan el proyecto, tanto la parte de la aplicación como la parte del servidor. En él también podremos encontrar otra documentación como: un manual de usuario de la aplicación, el código fuente de la misma y proposiciones de futuras versiones y mejoras de la aplicación actual.

Abstract

Don’t you know where you can eat a typical dish when you are on holidays? Do you want to go to eat to a different place? Do you want to enjoy that meal you love and you don’t know where you can do it? To answer those questions and thanks to the possibilities of modern smartphones’ technology, we present this application in my degree’s final project. This application, which runs with an Android operative system, gives us the option to find restaurants in our environment that offer the meal we really want. In addition, as a social network, it includes the option to rate the tasted dishes or to add new restaurants, giving the application versatility.

Nowadays our society is used to the use of smartphones and their possibilities. That is why we must to explore its potential to obtain better amenities. In the last few years the amount of available applications for these devices has increased too much, offering a huge variety of them. If we realize a research about their functionalities and uses we will discover that most of them are oriented to leisure. That is why we are going to start the inquiry of a software engineering project developing a restaurant localization restaurant for Android smartphones,

In this document you can find the UMI diagrams which model the project, both the application part and the server part. Besides, you can find other documents as: an application user manual, the source and proposals for future versions and improvements.

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación

Actualmente la sociedad está acostumbrada al uso diario de dispositivos móviles y las capacidades que estos nos brindan, por ello no es una cuestión baladí el intentar explotar todo su potencial para obtener comodidades. De un tiempo a esta parte, el número de aplicaciones disponibles para estos dispositivos ha ido en aumento haciendo que haya aplicaciones de todo tipo. Si realizamos un estudio de sus funcionalidades y usos, rápidamente comprobamos que, la mayor parte de estas están orientadas al ocio. Así pues, en base a esto, nos disponemos a realizar el estudio de un proyecto de ingeniería de software en el que desarrollaremos una aplicación de localización de restaurantes para dispositivos móviles con sistema operativo Android.

1.2. Funcionalidades

La aplicación que vamos a desarrollar tiene una funcionalidad clara que es la de localizar restaurantes en nuestro entorno que tengan una determinada comida. Pero además de esto posee varias funcionalidades más derivadas de esta. Ahora procederemos a enumerar las funcionalidades de la aplicación:

- Localización de restaurantes
  - Mediante gps, usando la posición actual del dispositivo.
  - Insertando una dirección en torno a la que buscar.
- Red social
  - Insertar nuevos restaurantes a la aplicación.
  - Puntuar platos de los restaurantes recomendados.

Mediante estas funcionalidades agregadas, se busca que la aplicación sea más completa, agregándole valores añadidos a la misma. El hecho de que se pueda buscar mediante gps o insertando una dirección facilita el uso de la aplicación, tanto para usos turísticos como para dispositivos que no posean gps. El hecho de que también posea las funcionalidades que hemos denominado como *red social* tendrá dos principales ventajas. La primera de ellas será el hecho de que el sistema podrá ir enriqueciéndose aumentando el número de restaurantes que se tendrán como referencia en las búsquedas. La segunda de estas ventajas será la capacidad de ofertar al usuario una información más completa y contrastada por otros usuarios a la hora de realizar consultas.

Capítulo 2

Documentación

2.1. Modelo del sistema

El sistema que vamos a utilizar para el desarrollo de la aplicación se basa en una estructura cliente-servidor. En este apartado vamos a estudiar la estructura que tendrá el sistema y como se interconectarán los distintos elementos del mismo.

2.1.1. Diagrama de bloques

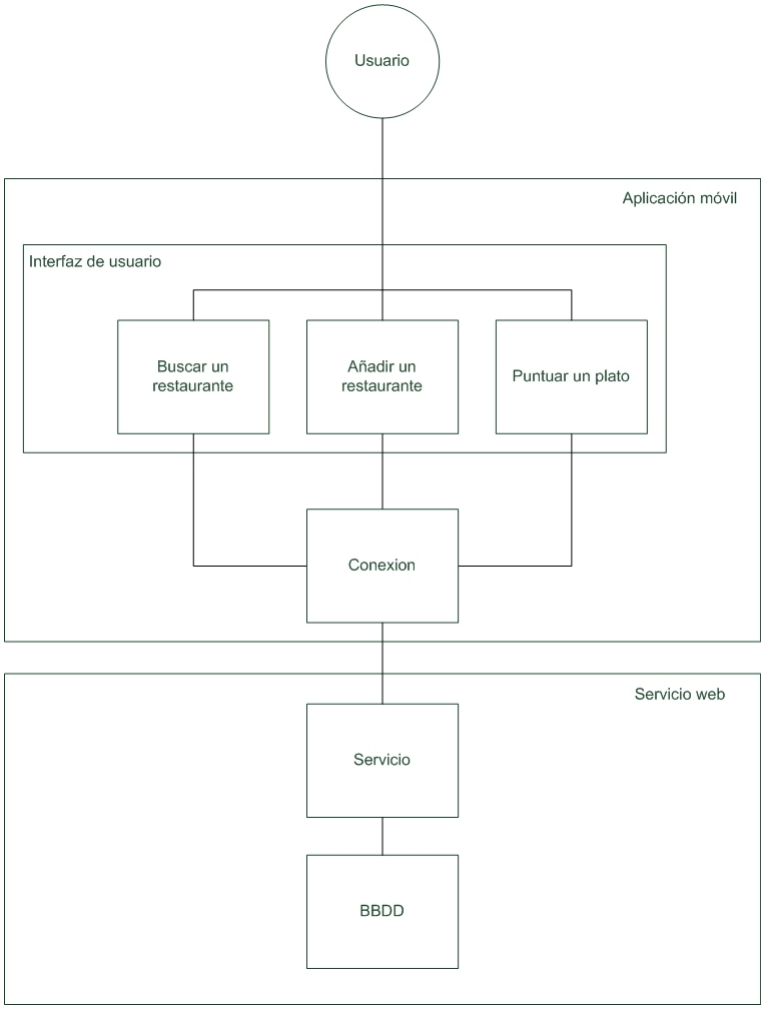

Figura 2.1: Diagrama de bloques

En esta primera aproximación al proyecto podemos observar dos bloques bien diferenciados: la aplicación móvil y el servidor. La parte del servidor posee el servicio propiamente dicho y la base de datos de la que hace uso. Por su parte, la aplicación tiene dos bloques separados claramente: la interfaz gráfica y el conector con el servicio.

Si nos detenemos a observar la parte de la interfaz gráfica de la aplicación móvil podemos comprobar que tiene tres bloques. Estos se corresponden con cada una de las funcionalidades básicas que tiene la aplicación.

2.1.2. Diagrama de despliegue

Mediante este diagrama podemos observar que la comunicación entre el cliente (aplicación móvil) y el servidor (servicio) se realizará mediante el protocolo HTTP. También podemos comprobar que el acceso a la base de datos se realizará mediante Java Database Connectivity (jdbc), un API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, que será el lenguaje escogido para el desarollo de todo el proyecto. Cabe destacar el hecho de que la base de datos es totalmente independiente de la aplicación móvil, lo que hace que sea transparente para esta.

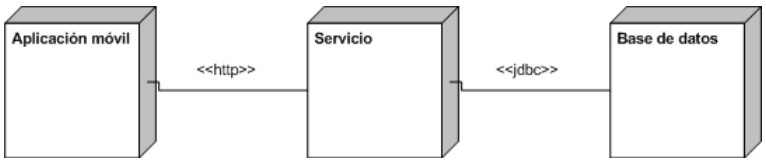

Figura 2.2: Diagrama de despliegue

2.1.3. Arquitectura del sistema

Si observamos la arquitectura de la aplicación de un modo más profundo podemos contemplar los componentes que tiene cada uno de los nodos. Cabe resaltar el componente serviOcio, presente tanto en la aplicación móvil como en el servicio. Este componente es el encargado de modelar los restaurantes y sus caracteristicas.

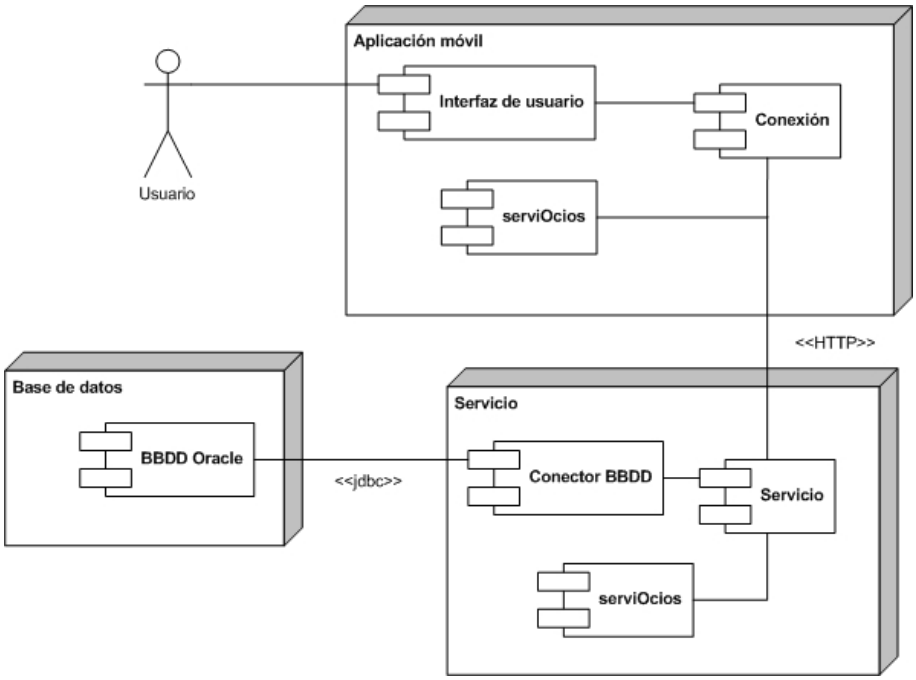

Figura 2.3: Diagrama de la arquitectura

2.2. Funcionalidad

En este apartado vamos a realizar un estudio de estas funciones que tiene la aplicación, estudiando tanto su flujo como los componentes que intervienen en ellas. Como ya hemos comentado previamente, la aplicación dispone de tres funcionalidades básicas:

- Buscar restaurantes.
- Añadir un restaurante.
- Puntuar un plato.

2.2.1. Casos de uso

Los casos de uso de la aplicación se corresponden con cada una de las funcionalidades que tiene la misma, así pues el diagrama de casos de uso será el siguiente.

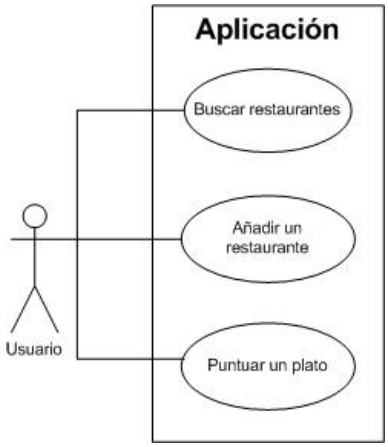

Figura 2.4: Diagrama de casos de uso

2.2.1.1. Buscar restaurantes

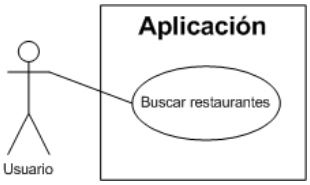

Figura 2.5: Diagrama de caso de uso: Buscar restaurantes

Esta funcionalidad nos permitirá realizar una búsqueda de restaurantes que dispongan entre sus platos la comida que indiquemos.

2.2.1.2. Añadir restaurantes

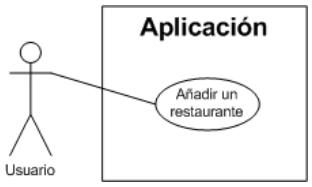

Figura 2.6: Diagrama de caso de uso: Añadir un restaurante

Como ya hemos visto, la aplicación se apoya en una base de datos. Esta funcionalidad nos permite añadir los datos de restaurantes a dicha base de datos.

2.2.1.3. Puntuar plato

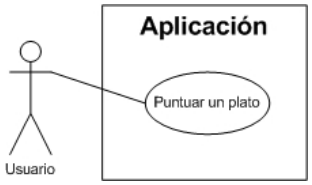

Figura 2.7: Diagrama de caso de uso: Puntuar un plato

Como acabamos de comentar, disponemos de una base de datos remota. Con esta funcionalidad podremos puntuar platos que hayamos pobado en un restaurante.

2.2.2. Diagramas de estados

Con estos diagramas representaremos el flujo de nuestra aplicación estudiando el diagrama de cada uno de los casos de uso y mostrando los estados por los que pasará. Para esto es conveniente que estudiemos lo que sucederá tanto en nuestra aplicación móvil como en el servidor por separado.

2.2.2.1. Buscar restaurantes: Aplicación móvil

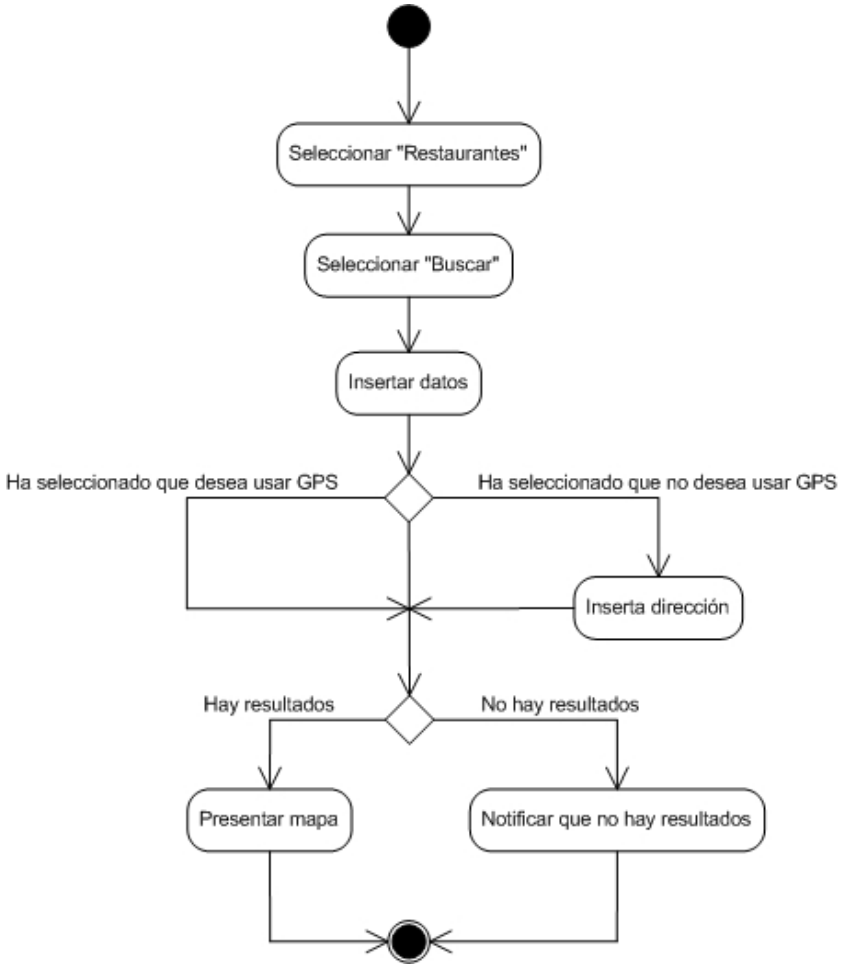

Figura 2.8: Diagrama de estados: Buscar restaurante (aplicación)

Para buscar un restaurante deberemos navegar por las distintas pantallas de la aplicación rellenando los datos pertinentes. En caso de que hayamos seleccionado que no queremos utilizar el gps, además de el nombre de la comida que deseemos, deberemos insertar una dirección. Con estos datos, se hará una llamada al servidor que nos responderá con los datos que se mostrarán por pantalla.

2.2.2.2. Buscar restaurantes: Servidor

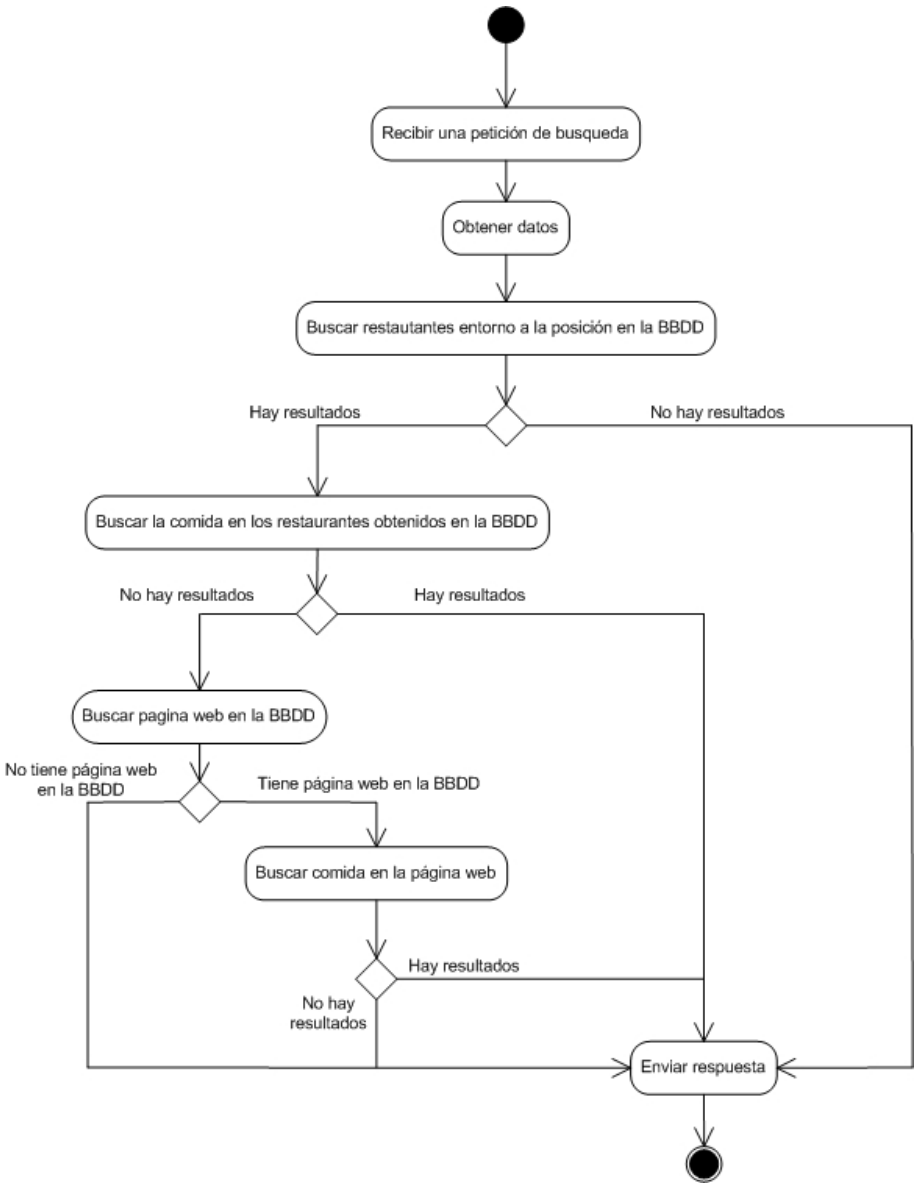

Figura 2.9: Diagrama de estados: Buscar restaurante (servidor)

Cuando el servidor recibe una petición de búsqueda de restaurantes lo primero que hace es obtener los datos de la petición. Con estos datos consulta la base de datos en busca de restaurantes que estén en torno a la posición recibida. Si hay restaurantes, buscará en la base de datos si tienen la comida deseada, de no ser así, y en caso de que los restaurantes tengan página web, buscará en la página web

la comida. Por último, los resultados obtenidos los mandará a la aplicación.

2.2.2.3. Añadir restaurantes: Aplicación móvil

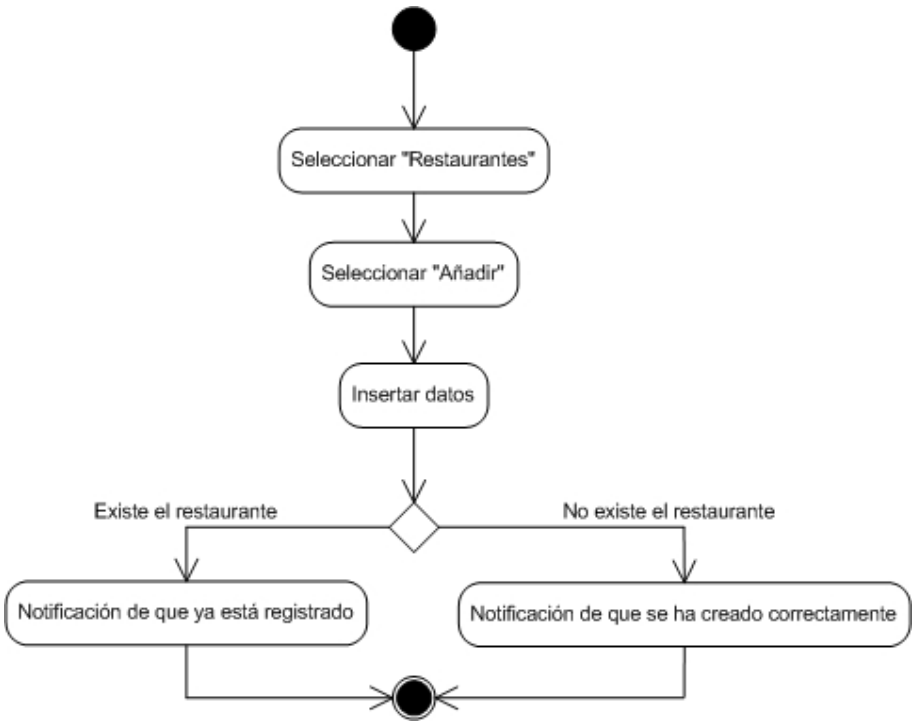

Figura 2.10: Diagrama de estados: Añadir restaurante (aplicación)

En el caso de que lo que deseemos sea añadir un restaurante, la aplicación nos pedirá los datos necesarios. Una vez cumplimentados correctamente se hará una llamada al servidor y con los datos devueltos se mostrará un mensaje de operación realizada correctamente o operación incorrecta.

2.2.2.4. Añadir restaurantes: Servidor

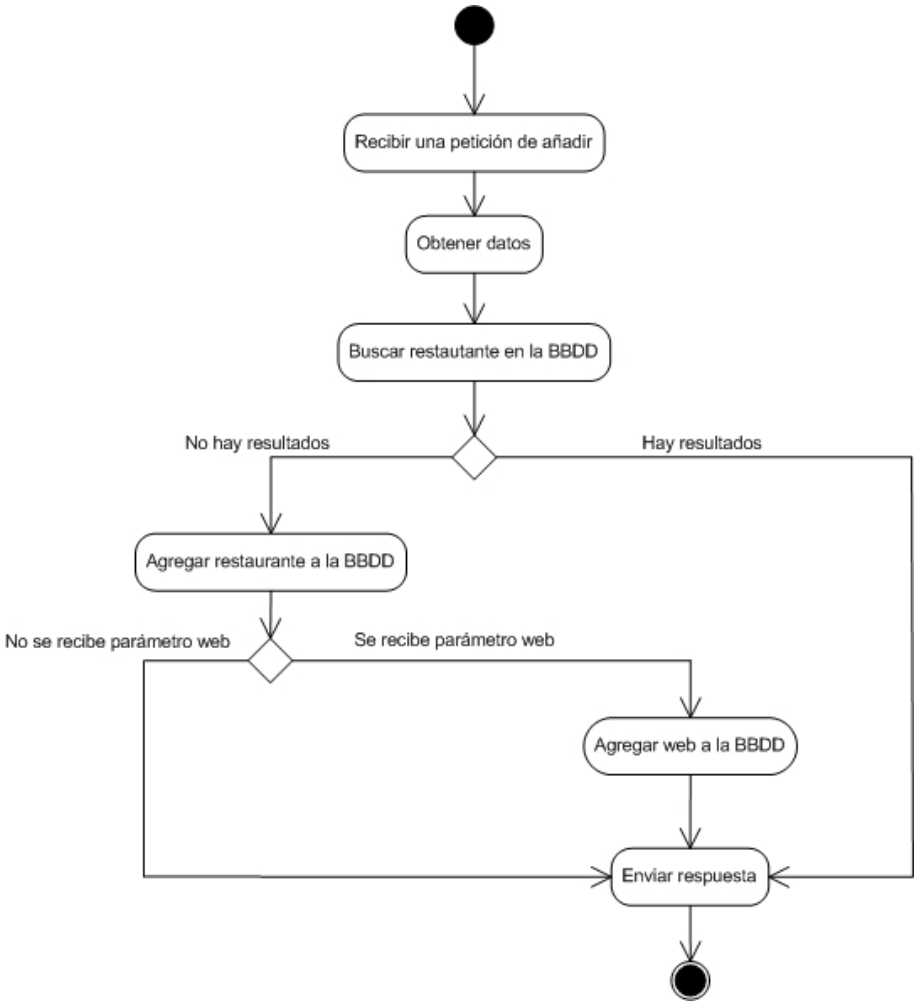

Figura 2.11: Diagrama de estados: Añadir restaurante (servidor)

Cuando el servidor recibe una petición para añadir un restaurantes, lo primero que hace es obtener los datos de la petición. Con estos datos consulta la base de datos para comprobar si existe ese restaurante. En caso de que no exista, lo añade y confirma la operación a la aplicación. En caso de existir, se lo comunica a la aplicación.

2.2.2.5. Puntuar plato: Aplicación móvil

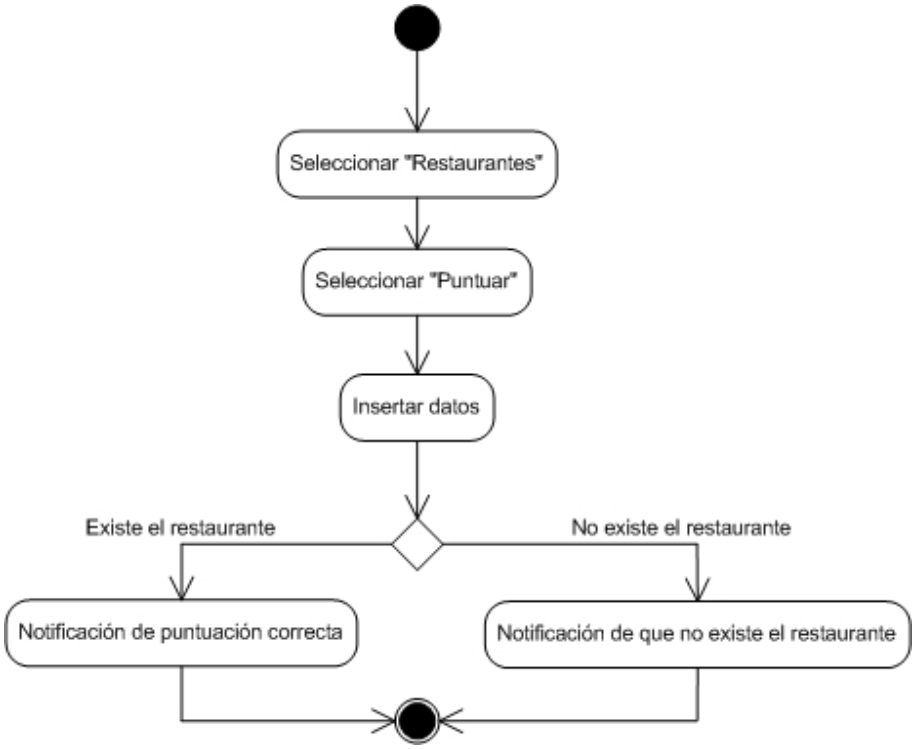

Figura 2.12: Diagrama de estados: Puntuar plato (aplicación)

Si lo que deseamos es puntuar un plato, la aplicación nos pedira que completemos los datos necesarios. Una vez completados correctamente, se realizará una petición al servidor. Según los datos que nos devuelva el servidor se mostrará un mensaje de error o de operación realizada.

2.2.2.6. Puntuar plato: Servidor

Figura 2.13: Diagrama de estados: Puntuar plato (servidor)

Cuando el servidor recibe una petición para puntuar un plato, obtiene los datos de la petición. Con estos consulta la base de datos para comprobar si existe el

restaurante al que pertenece el plato, si existe comprueba si se tiene ya puntuado el plato. De ser así, cambia la puntuación de este plato, recalcula la puntuación media del restaurante y manda una respuesta afirmativa. Si no existe el plato, crea el plato, lo puntúa, recalcula la media del restaurante y envía una respuesta afirmativa. Si el restaurante no existe, se manda un mensaje de error.

2.2.3. Diagramas de secuencia

Con estos diagramas podremos comprender como se ejecuta el proyecto cronológicamente. De nuevo tendremos un diagrama distinto para cada funcionalidad del sistema.

2.2.3.1. Buscar restaurantes

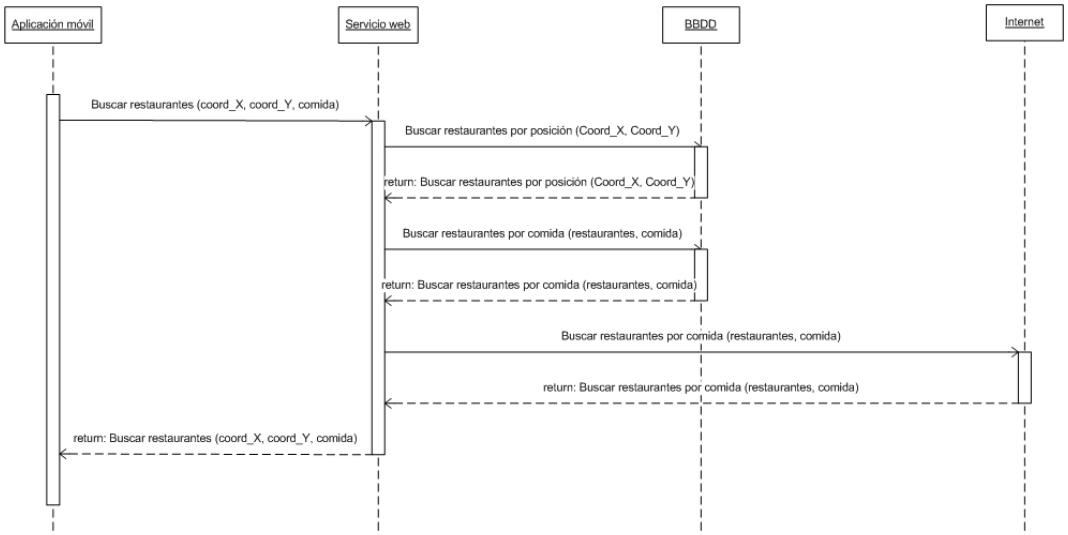

Figura 2.14: Diagrama de secuencia: Buscar restaurantes

Como podemos comprobar, nuestra aplicación móvil efectúa una única llamada al servicio web pasándole como parámetros las coordenadas de donde desea buscar y la comida que se desea. Es función del propio servicio web encargase de las sucesivas consultas a la base de datos e internet para obtener los resultados que se presentarán por la pantalla del dispositivo móvil. Cabe destacar que tanto si se utiliza gps, como si no, la llamada efectuada al servicio web es la misma, así pues debe ser función del dispositivo móvil el obtener las coordenadas de la dirección que inserte el usuario.

2.2.3.2. Añadir restaurantes

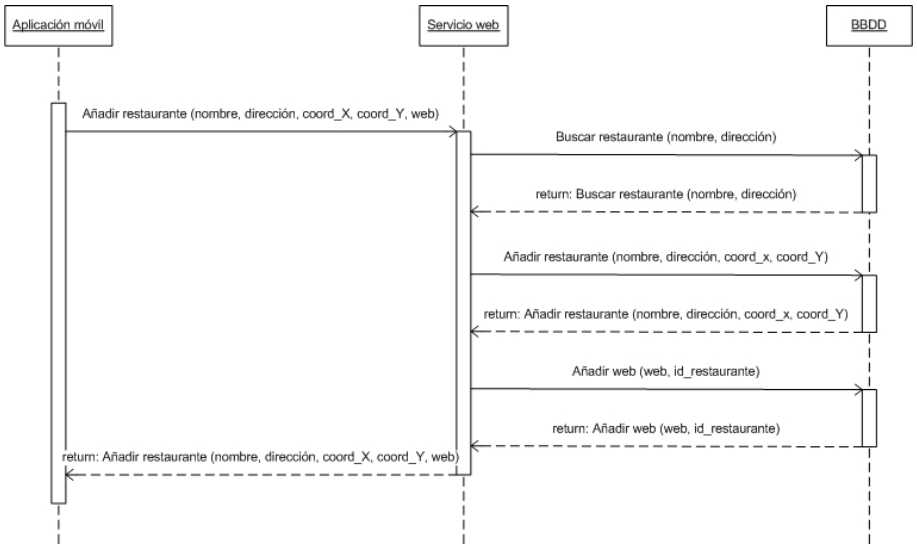

Figura 2.15: Diagrama de secuencia: Añadir restaurante

Al igual que sucedía anteriormente, la aplicación móvil solo efectúa una llamada al servicio web envíandole el nombre, la dirección, las coordenadas y la dirección web como parámetros. Al igual que en el caso previo, también es función de la propia aplicación móvil el obtener las coordenadas de la dirección insertada por el usuario, el resto de las comprobaciones en cuanto a si ya existe ese restaurante en la base de datos son realizadas por el servicio web.

2.2.3.3. Puntuar un plato

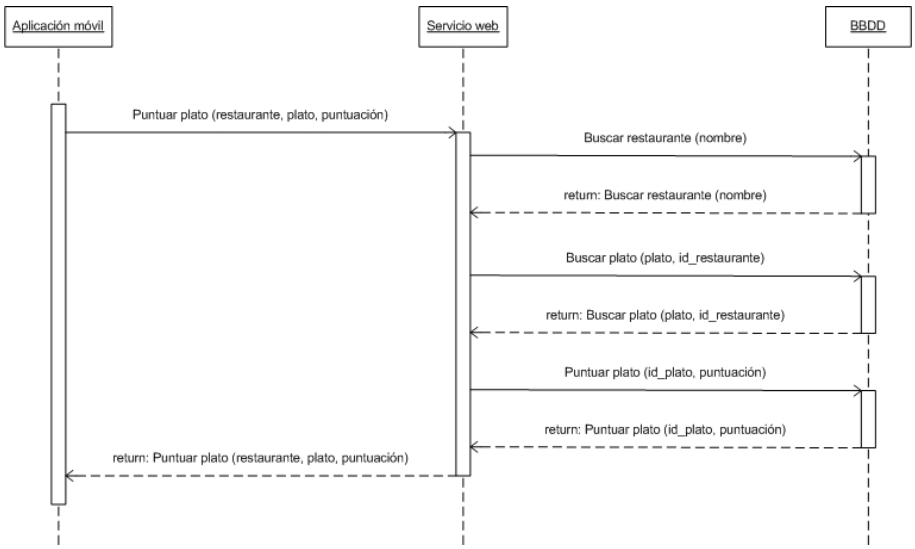

Figura 2.16: Diagrama de secuencia: Puntuar plato

En este caso, la aplicación móvil debe envíar los parámetros nombre, plato y puntuación al servicio web. Como en los casos anteriores, todas las comprobaciones de si existe ya ese plato registrado en la base de datos o si se debe crear son llevadas a cabo por el servicio web.

2.3. Diagrama de clases

En estos diagramas podremos ver más detenidamente como se ha implementado el proyecto. Como hemos estado realizando en otros puntos de este documento, a la hora de ver los diagramas de clases también separaremos la parte de la aplicación móvil y la parte del servidor. Pero previamente vamos a echar un vistazo rápido a todos los paquetes del proyecto y su relación:

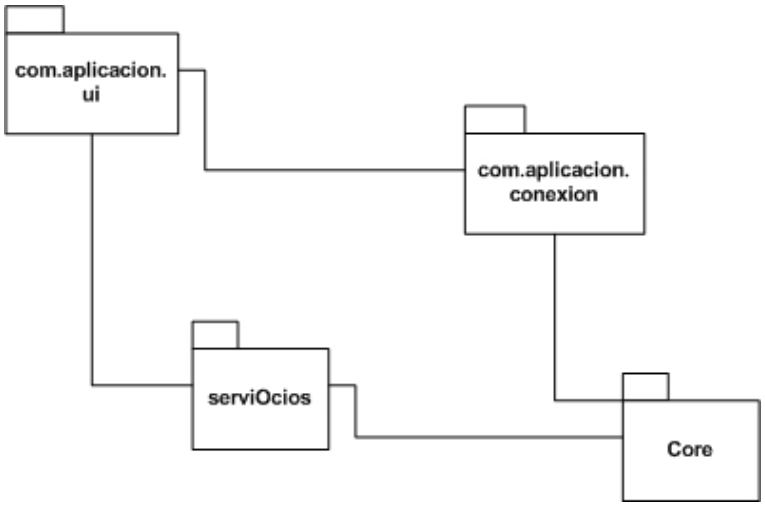

Figura 2.17: Diagrama de clases: Paquetes del proyecto

En este gráfico podemos ver como se distribuyen y relacionan los paquetes que forman parte del proyecto. En él podemos observar:

- **com.aplicacion.ui** Este paquete es el encargado de toda la presentación de menus y mapas en el dispositivo móvil.
- **com.aplicacion.conexion** Este paquete es el encargado de realizar la conexión entre el dispositivo móvil y el servidor mediante mensajes post.
- **Core** Es el core del servidor, se encarga de realizar las funciones del mismo.
- **serviOcios** Este paquete es el encargado de modelar todos los servicios que se vayan a implementar, en nuestro caso, restaurantes. Es un paquete que está presente tanto en el servidor como en el cliente.

Ahora procederemos a ver un diagrama de clases más detallado de cada uno de estos paquetes.

2.3.1. Diagrama de clases: com.aplicacion.ui

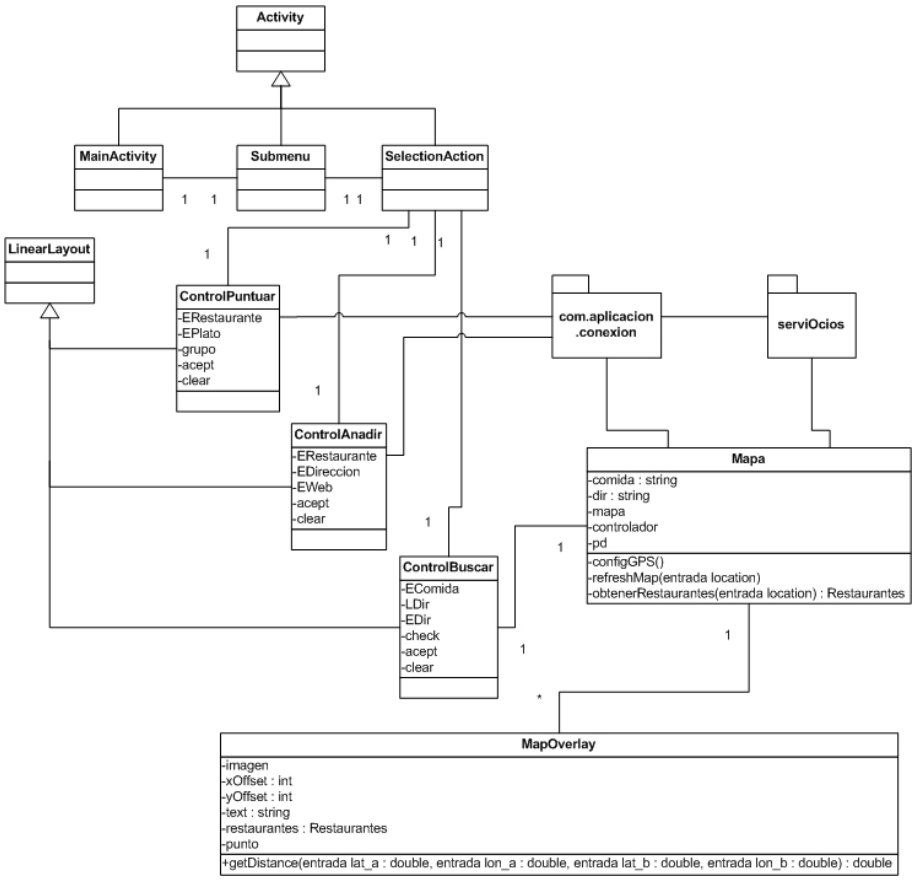

Figura 2.18: Diagrama de clases: com.aplicacion.ui

2.3.2. Diagrama de clases: com.aplicacion.conexion

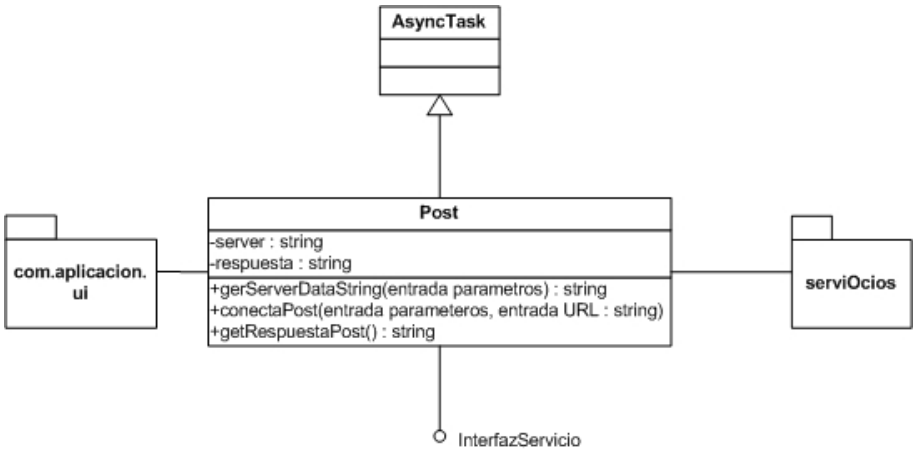

Figura 2.19: Diagrama de clases: com.aplicacion.conexion

2.3.3. Diagrama de clases: core

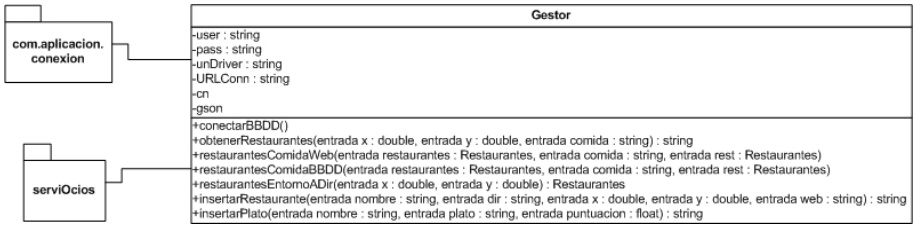

Figura 2.20: Diagrama de clases: core

2.3.4. Diagrama de clases: serviOcios

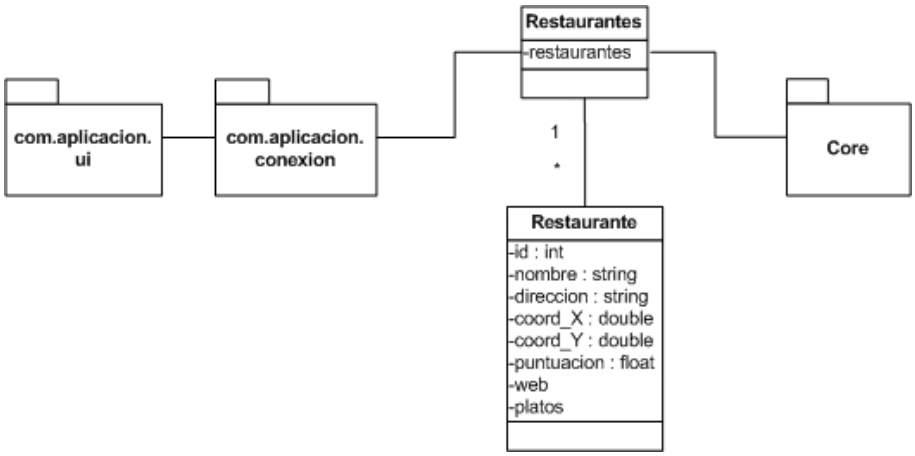

Figura 2.21: Diagrama de clases: serviOcios

2.4. Documentación de interfaces

2.4.1. Interfaz del servidor

Como ya comentamos previamente, la comunicación entre la aplicación web y el servidor es mediante en envío de mensajes POST del protocolo HTTP. Como hemos visto a lo largo de la documentación expuesta, nuestro servicio web posée tres funciones. A continuación procederemos a enumerar los parámetros que debemos enviar en nuestro mensaje post para el correcto funcionamiento de nuestro servidor.

2.4.1.1. Buscar restaurantes

Para hacer uso de esta funcionalidad nuestro mensaje post debe contener los siguientes parámetros:

- **option**: Identifica el tipo de operación a realizar. Para realizar una búsqueda de restaurantes debe ser '0'.
- **X**: Coordenada x (latitud).
- **Y**: Coordenada y (longitud).
- **comida**: Comida que se desea buscar.
- **OK**: Indica la validez de la petición. Debe existir este parámetro en el mensaje post y ser distinto de null.

2.4.1.2. Añadir restaurante

Para hacer uso de esta funcionalidad nuestro mensaje post debe contener los siguientes parámetros:

- **option**: Identifica el tipo de operación a realizar. Para realizar una búsqueda de restaurantes debe ser '1'.
- **nombre**: Nombre del restaurante que se desea añadir.
- **dir**: Dirección del restaurante que se desea añadir.
- **X**: Coordenada x (latitud) del restaurante que se desea añadir.
- **Y**: Coordenada y (longitud) del restaurante que se desea añadir.
- **web**: Página web del restaurante que se desea añadir.

- **OK**: Indica la validez de la petición. Debe existir este parámetro en el mensaje post y ser distinto de null.

2.4.1.3. Puntuar plato

Para hacer uso de esta funcionalidad nuestro mensaje post debe contener los siguientes parámetros:

- **option**: Identifica el tipo de operación a realizar. Para realizar una búsqueda de restaurantes debe ser '2'.
- **nombre**: Nombre del restaurante en el que se desea puntuar el plato.
- **plato**: Nombre del plato que se desea puntuar.
- **puntuacion**: Puntuación que se le da al plato.
- **OK**: Indica la validez de la petición. Debe existir este parámetro en el mensaje post y ser distinto de null.

2.5. Documentación de la base de datos

Como hemos estado viendo a lo largo de toda la documentación, el servicio web hace uso de una base de datos. En este apartado nos disponemos a ver un poco más en profuncidad el diseño de la misma.

2.5.1. Tablas

restaurante:

id	nombre	direccion	coor_X	coor_Y	puntuacion

- **id**: Se trata del identificador de los restaurantes de tipo entero. Es de tipo integer, único, autoincremental y no puede ser nulo.
- **nombre**: Es una cadena de sesenta caracteres como máximo. Es el nombre del restaurante. No puede ser nulo.
- **direccion**: Es una cadena de cien caracteres como máximo. Es la dirección del restaurante. No puede ser nulo.
- **coor_X**: Se trata de la coordenada x del restaurante (latitud). Es del tipo float.
- **coor_Y**: Se trata de la coordenada y del restaurante (longitud). Es del tipo float.
- **puntuacion**: Es la puntuación media de todos los platos que se tiene registrados y puntuados del restaurante. Si no hay puntuaciones es -1. Es del tipo float.

web:

id	uri	id_restaurante

- **id**: Se trata del identificador de las direcciones web. Es de tipo integer, único, autoincremental y no puede ser nulo.
- **uri**: Es una cadena de doscientos caracteres como máximo. Es el la dirección web del restaurante. No puede ser nulo.
- **id_restaurante**: Es el indice de restaurante al que pertenede la web. Es clave foranea, así pues, no puede ser nulo. Es del tipo integer.

plato:

id	plato	calificación	id_restaurante

- **id**: Se trata del identificador de los platos. Es de tipo integer, único, autoincremental y no puede ser nulo.
- **plato**: Es una cadena de sesenta caracteres como máximo. Es el nombre del plato. No puede ser nulo.
- **calificacion**: Es la calificación del plato. Es del tipo float.
- **id_restaurante**: Es el indice de restaurante al que pertenede el plato. Es clave foranea, así pues, no puede ser nulo. Es del tipo integer.

2.5.2. Modelo relacional

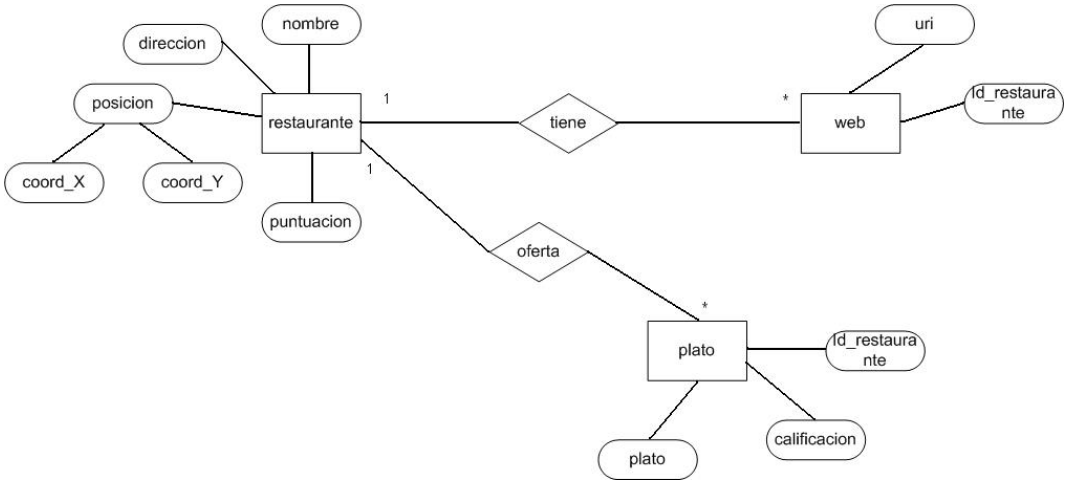

Figura 2.22: Modelo relacional de la base de datos

Capítulo 3

Versiones y mejoras

Tras la presentación de este proyecto, nos planteamos nuevos usos y mejoras para la aplicación como:

- Añadir capacidades para buscar películas de cine.
- Añadir capacidades para buscar obras de teatro.
- Optimizar interfaz gráfica.
- Realizar un registro de usuarios.
- Realizar funciones batch en el servidor que comprueben la consistencia de los datos de la base de datos.

Apéndice A

Manual de usuario

Con esta aplicación podrá localizar de un modo sencillo restaurantes que dispongan de la comida que busque tanto a su alrededor como en torno a una dirección que indique. Además, podrá añadir restaurantes y puntuar los platos que ha degustado, haciendo que, tanto usted como otros usuarios se beneficien de las ventajas que esto supone. En este pequeño manual vamos a explicar paso a paso como sacar el máximo rendimiento a esta aplicación.

A.1. Iniciando nuestro uso en serviOcios

Cuando inicie la aplicación, encontrará una pantalla en la cual podrá seleccionar qué tipo de alternativa de ocio desea buscar. Actualmente solo está disponible la opción *Restaurantes*.

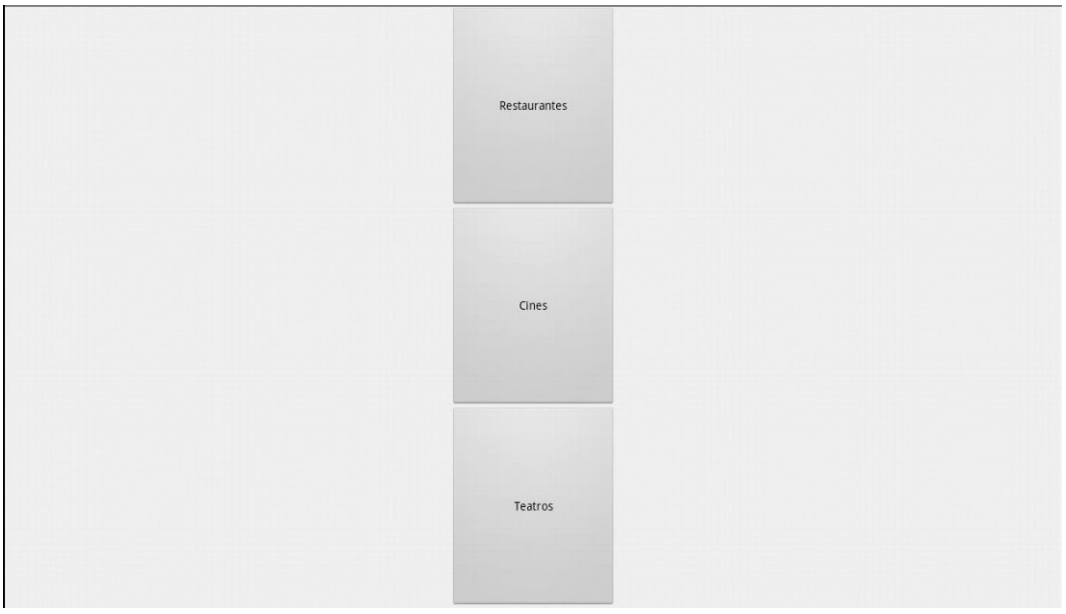

Figura A.1: Pantalla inicial

Esto le conducirá a un submenú en el que tendrá varias alternativas para realizar: buscar restaurantes, añadir un restaurante o puntuar un plato.

Figura A.2: Submenu

A.2. Buscando restaurantes

Si la opción deseada es la de localizar restaurantes mediante nuestra aplicación, usted podrá buscar restaurantes que contengan en su carta un plato que desée tomar. Esta funcionalidad tiene dos módos de uso:

- Mediante gps
- Indicando una dirección

Para poder acceder a cualquiera de los dos módos de búsqueda, deberá pulsar la opción *Buscar restaurantes* del submenu.

Una vez haya puldado dicha opción, aparecerá un formulario en el que tendrá que indicar qué desea comer y si usará gps o no. En caso de que seleccione que no desea usar gps, le aparecerá un nuevo campo para insertar la dirección en torno a la que desea buscar.

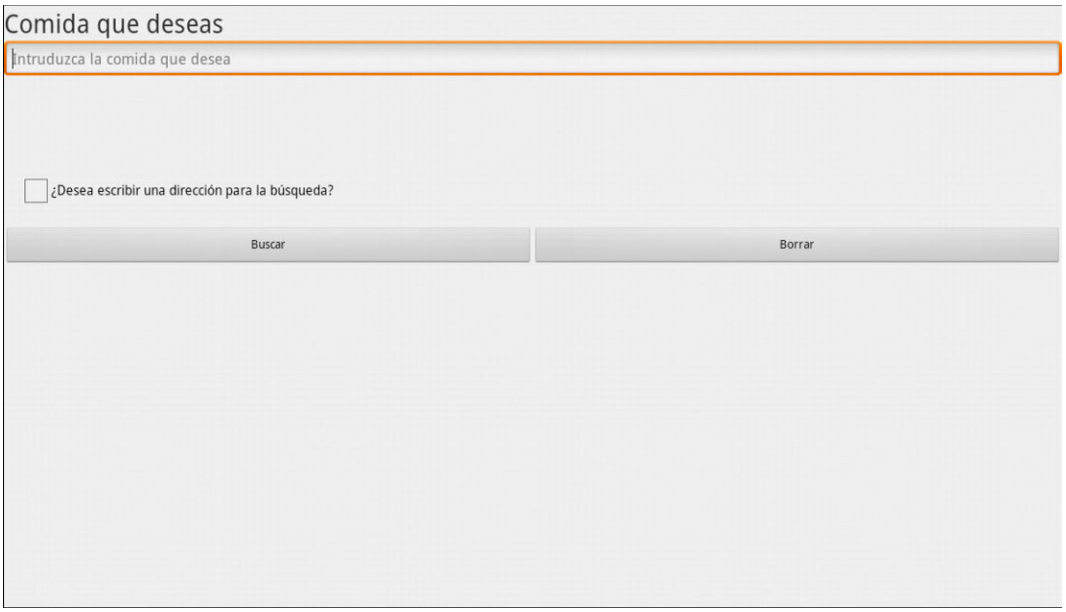

Figura A.3: Formulario usando gps

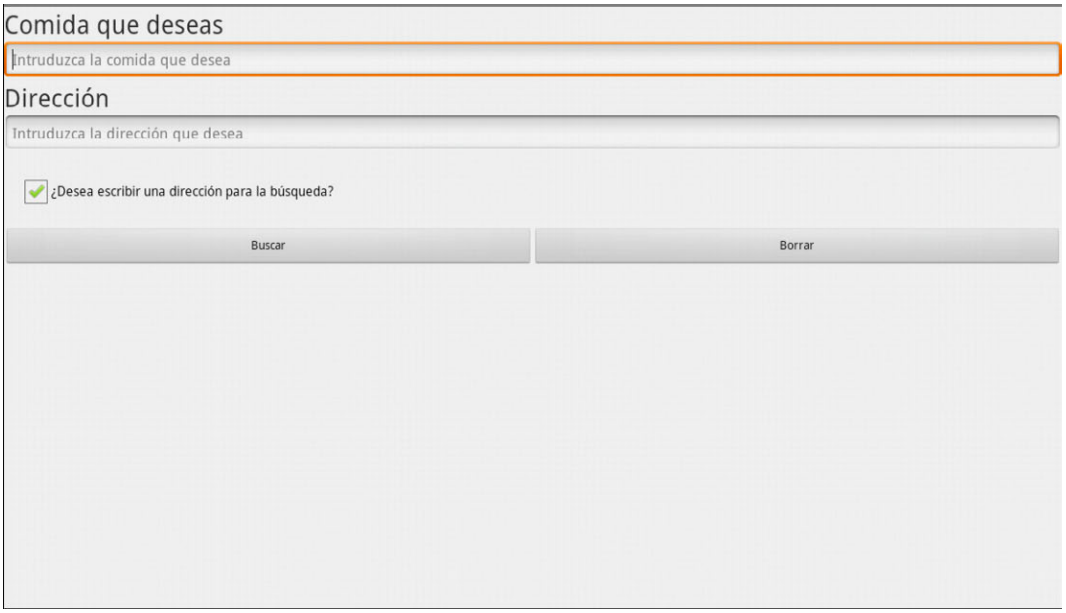

Figura A.4: Formulario sin el uso de gps

Después de rellenar el formulario y pulsar el botón de buscar, en ambos casos, y tras un tiempo de espera, le aparecerán los resultados dibujados en un mapa.

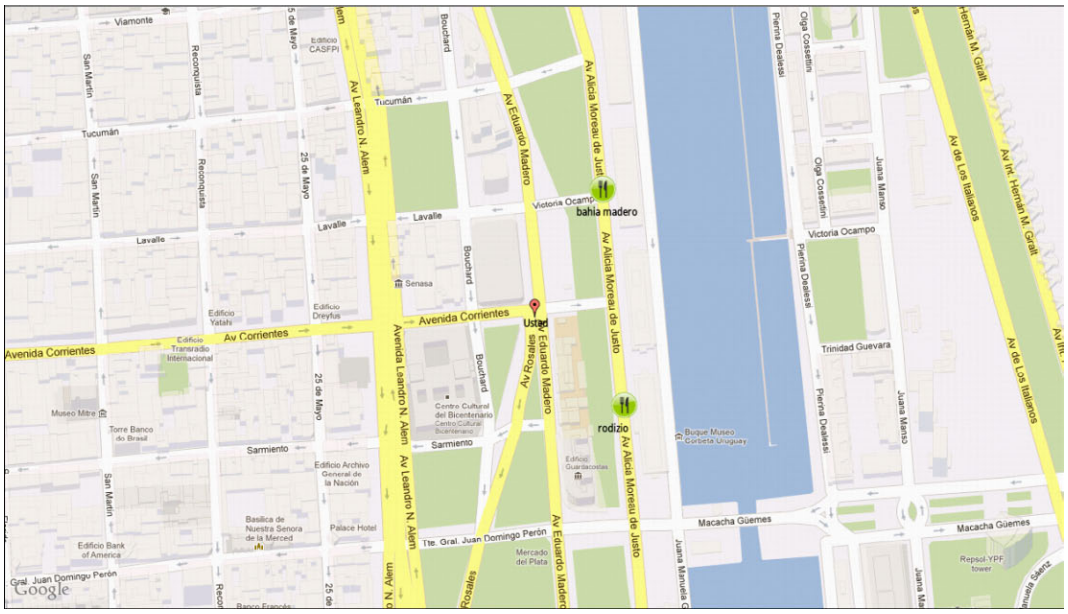

Figura A.5: Resultado de una consulta

En este mapa, podrá pulsar en los iconos de los restaurantes para obtener más información sobre ellos como su nombre, dirección y puntuación dada por los usuarios de la aplicación.

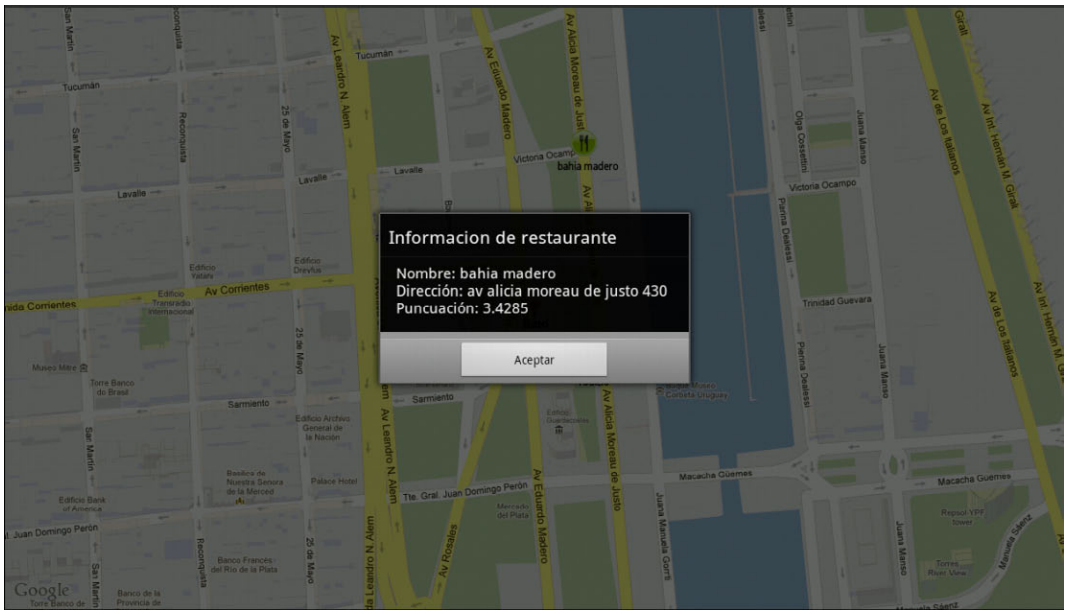

Figura A.6: Detalle de un restaurante

De este modo, podremos saber que restaurantes disponen de esa comida que deseamos.

A.3. Añadir restaurantes

Como se comentó previamente, esta aplicación también es capaz de aprender nuevos restaurantes mediante la colaboración de los usuarios, así pues, ahora procederemos a ver cómo se pueden añadir nuevos restaurantes.

Para ello, en la pantalla del submenu deberá seleccionar la opción *Añadir un restaurante*, esto le conducirá a un nuevo formulario en el cual podrá insertar los datos de un restaurante.

Los campos Nombre y Dirección son obligatorios, así pues, en caso de que no los cumplimente aparecerá un aviso por pantalla indicándo que deben ser rellenados. Cabe destacar que cuanto más exacto sea a la hora de poner la dirección, mayor facilidad y enriquecimiento habrá entre los usuarios, por ejemplo: poner *Corrientes 123* puede llevar a equívocos y es menos exácto que poner *av Corrientes 123*, incluso, si quisieramos ser más estrictos, podríamos poner *av Corrientes 123 Buenos Aires*, haciendo así que no haya duda alguna de la localización del restaurante.

Figura A.7: Mensaje de alta correcta de un restaurante

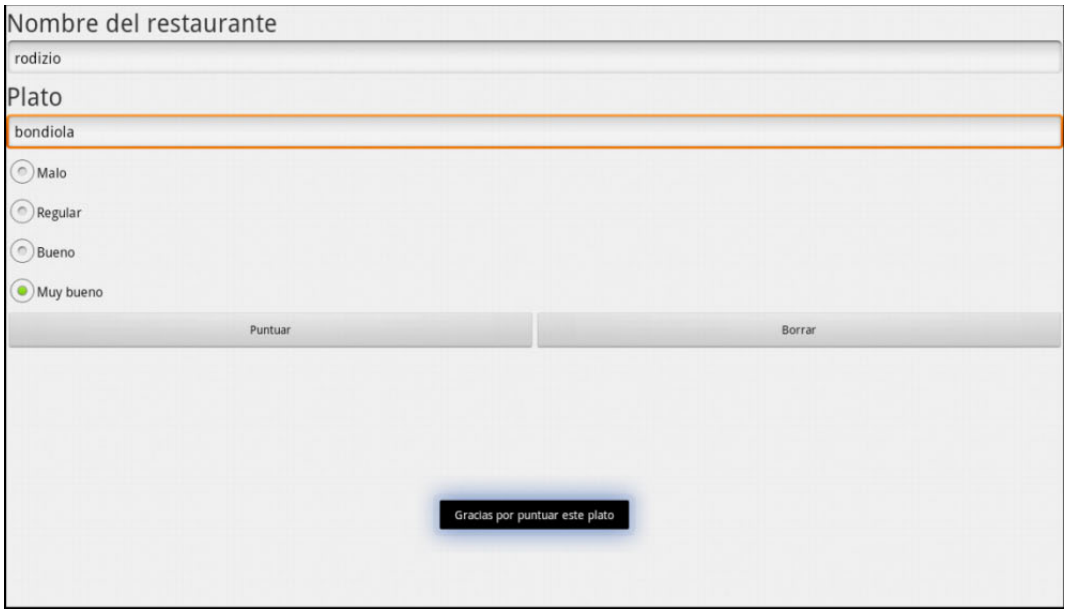

Figura A.8: Formulario para puntuar un plato

A.4. Puntuar un plato

La puntuación que tiene cada uno de los restaurantes, viene dada por la media de la puntuación de los platos que tiene registrado, así pues, este paso es fundamental para que la aplicación pueda aportar mayor información en cuanto a la calidad de los restaurantes. Para poder puntuar un plato, deberá pulsar la opción *Puntuar plato* del submenu. Esto le conducirá a un formulario en el cual deberá completar el nombre del restaurante, del plato y la puntuación que desée darle. En caso de que el restaurante no exista, la aplicación dará un mensaje de error invitandole a agregar ese restaurante a la aplicación.

Con estos simples consejos, ya está en disposicón de sacar el mayor rendimiento a la aplicación.

¡Buen provecho!

Apéndice B

Código fuente

B.1. Código java

B.1.1. Paquete com.aplicacion.ui

Clase MainActivity.java

```
/*****************************************************
 *
 * Fichero: MainActivity.java
 * Clase que modela la pantalla inicial de la aplicacion
 * @author Eduardo Campos de Diago
 * @version 1.2
 *
 *****************************************************/

package com.aplicacion.ui;


import android.os.Bundle;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.DialogInterface.OnClickListener;
import android.view.View;
import android.widget.Button;

public class MainActivity extends Activity
{

/*****************************************************
 * Es llamado cuando es creado el primer activity
 *
 *
 *****************************************************/
 @Override
 public void onCreate(Bundle savedInstanceState)
 {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

```

```
37          Button bRestaurantes = (Button)findViewById(R.id.
               bot_restaurantes);
38          Button bCines = (Button)findViewById(R.id.bot_cines
               );
39          Button bTeatros = (Button)findViewById(R.id.
               bot_teatros);
40
41          //creaci n de los Listener
42          final View.OnClickListener eventoRestaurantes = new
                View.OnClickListener()
43          {
44                  public void onClick(View v)
45                  {
46                          Intent intent = new Intent (
                               MainActivity.this,
47                                   Submenu.class);
48                          Bundle bundle = new Bundle();
49                          bundle.putShort("Tipo", (short)1);
50                          intent.putExtras(bundle);
51                          startActivity(intent);
52                  }
53          };
54
55          final View.OnClickListener eventoCines = new View.
               OnClickListener()
56          {
57                  public void onClick(View v)
58                  {
59                          /*Intent intent = new Intent (
                               MainActivity.this,
60                                           Submenu.class);
61                          Bundle bundle = new Bundle();
62                          bundle.putShort("Tipo", (short)2);
63                          intent.putExtras(bundle);
64                          startActivity(intent);*/
65                          AlertDialog.Builder builder = new
                               AlertDialog.Builder(context);
66
67                          builder.setTitle("Informacion
                               serviOcios");
68                          builder.setMessage("Esta opci n no
                                est  disponible a n");
```

```
69                          builder.setPositiveButton("Aceptar
                               ", new OnClickListener(){
70                                  public void onClick(
                                       DialogInterface dialog,
                                       int which) {
71                                              dialog.
                                                   cancel()
                                                   ;
72                                      }
73                                  });
74                          builder.create();
75                          builder.show();
76                  }
77          };
78
79          final View.OnClickListener eventoTeatros = new View
               .OnClickListener()
80          {
81                  public void onClick(View v)
82                  {
83                          /*Intent intent = new Intent (
                               MainActivity.this,
84                                      Submenu.class);
85                          Bundle bundle = new Bundle();
86                          bundle.putShort("Tipo", (short)3);
87                          intent.putExtras(bundle);
88                          startActivity(intent);*/
89                          AlertDialog.Builder builder = new
                               AlertDialog.Builder(context);
90
91                          builder.setTitle("Informacion
                               serviOcios");
92                          builder.setMessage("Esta opci n no
                                est  disponible a n");
93                          builder.setPositiveButton("Aceptar
                               ", new OnClickListener(){
94                                  public void onClick(
                                       DialogInterface dialog,
                                       int which) {
95                                              dialog.
                                                   cancel()
                                                   ;
96                                      }
```

```
97                                      });
98                              builder.create();
99                              builder.show();
100                     }
101             };
102
103             //asociaci n de adaptadores y eventos
104             bRestaurantes.setOnClickListener(eventoRestaurantes
                    );
105             bCines.setOnClickListener(eventoCines);
106             bTeatros.setOnClickListener(eventoTeatros);
107
108     }
109 }
```

Clase Submenu.java

```
/*****************************************************
 *
 * Fichero: Submenu.java
 * Clase que caracteriza el menu para la eleccion de accion
 * @author Eduardo Campos de Diago
 * @version 1.2
 *
 *****************************************************/

package com.aplicacion.ui;


import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.AdapterView.OnItemClickListener;

public class Submenu extends Activity
{
/*****************************************************
 * Es llamado cuando es creado
 *
 *
 *****************************************************/

 public void onCreate(Bundle savedInstanceState)
 {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.submenu);

        String[] datoList = new String[]{"Buscar","A adir
           un restaurante", "Puntuar un plato"};

        Bundle bundle = getIntent().getExtras();
        short option =  bundle.getShort("option");

        //Creaci n del ListView
```

```
41          ArrayAdapter<String> adapterList = new ArrayAdapter
               <String>(this, android.R.layout.
               simple_list_item_1, datoList);
42          ListView list = (ListView)findViewById(R.id.list);
43          list.setAdapter(adapterList);
44
45          //Adaptador del ListView y del GridView
46                  OnItemClickListener eventoList = new
                       OnItemClickListener()
47                  {
48
49                          public void onItemClick(AdapterView
                               <?> parent, View v, int position
                               , long id)
50                          {
51                                  Intent intent = new Intent
                                       (Submenu.this,
                                       SelectionAccion.class);
52                          Bundle bundle = new Bundle();
53                          bundle.putShort("option", (short)
                               parent.getItemIdAtPosition(
                               position));
54                          intent.putExtras(bundle);
55                          startActivity(intent);
56                          }
57                  };
58
59                  //asociaci n de adaptadores y eventos
60                  list.setOnItemClickListener(eventoList);
61    }
62  }
```

Clase SelectionAction.java

```
/*****************************************************
 *
 * Fichero: SelectionAction.java
 * Clase que caracteriza el menu para la eleccion de accion
 * @author Eduardo Campos de Diago
 * @version 1.2
 *
 *****************************************************/

package com.aplicacion.ui;


import android.app.Activity;
import android.os.Bundle;

public class SelectionAccion extends Activity
{

/*****************************************************
 * Es llamado cuando es creado
 *
 *
 *****************************************************/
 public void onCreate(Bundle savedInstanceState)
 {
        super.onCreate(savedInstanceState);

        Bundle bundle = getIntent().getExtras();
        short option =  bundle.getShort("option");

        switch (option)
        {
                case 0:
                {
                        setContentView(R.layout.
                           layout_buscar);
                        break;
                }
                case 1:
                {
```

```
40                              setContentView(R.layout.
                                   layout_anadir);
41                              break;
42                      }
43                      case 2:
44                      {
45                              setContentView(R.layout.
                                   layout_puntuar);
46                              break;
47                      }
48              }
49      }
50  }
```

Clase ControlBuscar.java

```
/*****************************************************
 *
 * Fichero: ControlBuscar.java
 * Clase que modela la pantalla de la funcion buscar de la
    aplicacion
 * @author Eduardo Campos de Diago
 * @version 1.2
 *
 *****************************************************/

package com.aplicacion.ui;


import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.CompoundButton.
   OnCheckedChangeListener;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.util.AttributeSet;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.EditText;


public class ControlBuscar extends LinearLayout
{
 private EditText EComida;
 private TextView LDir;
 private EditText EDir;
 private CheckBox check;
 private Button acept;
 private Button clear;

/*****************************************************
 * Constructor de la clase ControlBuscar
 *
```

```
40   * @param Context contesto de la aplicacion
41   *
42   ****************************************************/
43   public ControlBuscar(Context context)
44   {
45          super(context);
46          initialize();
47   }
48
49  /****************************************************
50   * Constructor de la clase ControlBuscar
51   *
52   * @param Context contesto de la aplicacion
53   * @param AttributeSet Atributos
54   *
55   ****************************************************/
56   public ControlBuscar(Context context, AttributeSet attrs)
57   {
58          super(context, attrs);
59          initialize();
60   }
61
62  /****************************************************
63   * Inicializa la interfaz para ser mostrada
64   *
65   ****************************************************/
66   private void initialize()
67   {
68          String interfaz = Context.LAYOUT_INFLATER_SERVICE;
69          LayoutInflater inflater = (LayoutInflater)
               getContext().getSystemService(interfaz);
70          inflater.inflate(R.layout.buscar, this, true);
71
72          EComida = (EditText)findViewById(R.id.edit_comida);
73          LDir = (TextView)findViewById(R.id.label_direccion)
               ;
74          EDir = (EditText)findViewById(R.id.edit_direccion);
75          check = (CheckBox)findViewById(R.id.checkBox_gps);
76          acept = (Button)findViewById(R.id.button_Buscar);
77          clear = (Button)findViewById(R.id.button_Clear);
78
79          addEvents();
80   }
```

```
81
82  /*****************************************************
83   * Anade los eventos a los botones de la pantalla
84   *
85   *****************************************************/
86   private void addEvents()
87   {
88          final OnClickListener eventoAcept = new
               OnClickListener()
89          {
90                  public void onClick(View v)
91                  {
92                          Intent intent = new Intent (
                               getContext(), Mapa.class);
93                          Bundle bundle = new Bundle();
94                          bundle.putString("comida",
                               getComida());
95                          bundle.putString("dir", getDir());
96                          intent.putExtras(bundle);
97                          getContext().startActivity(intent);
98                  }
99          };
100
101         final OnClickListener eventoClear = new
               OnClickListener()
102         {
103                 public void onClick(View v)
104                 {
105                         EComida.setText("");
106                         EDir.setText("");
107                         check.setChecked(false);
108                 }
109         };
110
111         final OnCheckedChangeListener eventoCheck = new
               OnCheckedChangeListener()
112         {
113
114                 public void onCheckedChanged(CompoundButton
                        buttonView, boolean isChecked)
115                 {
116                         if (isChecked)
117                         {
```

```
118                                     LDir.setVisibility(VISIBLE)
                                           ;
119                                     EDir.setVisibility(VISIBLE)
                                           ;
120                             }
121                             else
122                             {
123                                     EDir.setText("");
124                                     LDir.setVisibility(
                                           INVISIBLE);
125                                     EDir.setVisibility(
                                           INVISIBLE);
126                             }
127
128                     }
129             };
130
131             acept.setOnClickListener(eventoAcept);
132             clear.setOnClickListener(eventoClear);
133             check.setOnCheckedChangeListener(eventoCheck);
134     }
135
136     /*****************************************************
137      * Obtiene el valor del string escrito en la etiqueta de
           comida
138      *
139      * @return String comida que se busca
140      *
141      *****************************************************/
142     public String getComida()
143     {
144             return EComida.getText().toString();
145     }
146
147     /*****************************************************
148      * Obtiene el valor del string escrito en la etiqueta de
           direccion
149      *
150      * @return String direccion donde buscar
151      *
152      *****************************************************/
153     public String getDir()
154     {
```

```
155             return EDir.getText().toString();
156     }
157   }
```

Clase ControlAnadir.java

```
/*****************************************************
 *
 * Fichero: ControlAnadir.java
 * Clase que modela la pantalla de la funcion anadir de la
    aplicacion
 * @author Eduardo Campos de Diago
 * @version 1.2
 *
 *****************************************************/

package com.aplicacion.ui;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.ExecutionException;

import com.aplicacion.conexion.Post;

import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.Toast;
import android.annotation.SuppressLint;
import android.content.Context;
import android.location.Address;
import android.location.Geocoder;
import android.util.AttributeSet;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.EditText;


public class ControlAnadir extends LinearLayout
{
        private EditText ERestaurante;
        private EditText EDireccion;
        private EditText EWeb;
        private Button acept;
        private Button clear;

/*****************************************************
```

```
41   * Constructor de la clase ControlAnadir
42   *
43   * @param Context contesto de la aplicacion
44   *
45   ****************************************************/
46   public ControlAnadir(Context context)
47   {
48          super(context);
49          initialize();
50   }
51
52  /****************************************************
53   * Constructor de la clase ControlAnadir
54   *
55   * @param Context contesto de la aplicacion
56   * @param AttributeSet Atributos
57   *
58   ****************************************************/
59   public ControlAnadir(Context context, AttributeSet attrs)
60   {
61          super(context, attrs);
62          initialize();
63   }
64
65  /****************************************************
66   * Inicializa la interfaz para ser mostrada
67   *
68   ****************************************************/
69   private void initialize()
70   {
71          String interfaz = Context.LAYOUT_INFLATER_SERVICE;
72          LayoutInflater inflater = (LayoutInflater)
               getContext().getSystemService(interfaz);
73          inflater.inflate(R.layout.anadir, this, true);
74
75          ERestaurante = (EditText)findViewById(R.id.
               edit_restaurante);
76          EDireccion = (EditText)findViewById(R.id.
               edit_direccion);
77          EWeb = (EditText)findViewById(R.id.edit_web);
78          acept = (Button)findViewById(R.id.button_Anadir);
79          clear = (Button)findViewById(R.id.button_Clear);
80
```

```
81          addEvents();
82   }
83
84  /*****************************************************
85   * Anade los eventos a los botones de la pantalla
86   *
87   *****************************************************/
88   private void addEvents()
89   {
90          final OnClickListener eventoAcept = new
               OnClickListener()
91          {
92                  @SuppressWarnings("unchecked")
93                  @SuppressLint("ShowToast")
94                  public void onClick(View v)
95                  {
96                  try
97                  {
98                  if (getRestaurante().length() < 1 ||
                       getDireccion().length() < 1)
99                  {
100                         Toast.makeText(getContext(), "Debes
                                completar tanto el nombre del
                               restaurante como su direcci n",
                                Toast.LENGTH_LONG).show();
101                 }
102                 else
103                 {
104                         ArrayList<String> parametros = new
                               ArrayList<String>();
105
106                         parametros.add("option");
107                         parametros.add("1");
108                         parametros.add("nombre");
109                         parametros.add(getRestaurante().
                               toLowerCase());
110                         parametros.add("dir");
111                         parametros.add(getDireccion().
                               toLowerCase());
112                         parametros.add("web");
113                         parametros.add(getWeb());
114
```

```
115                         Geocoder geocoder = new Geocoder(
                               getContext());
116
117                         List<Address> addresses;
118
119                         addresses = geocoder.
                               getFromLocationName(getDireccion
                               (), 1);
120
121                         if(addresses.size() > 0)
122                         {
123                                 parametros.add("X");
124                                 double longitude =
                                       addresses.get(0).
                                       getLongitude();
125                                 parametros.add(String.
                                       valueOf(longitude));
126                                 parametros.add("Y");
127                                 double latitude = addresses
                                       .get(0).getLatitude();
128                                 parametros.add(String.
                                       valueOf(latitude));
129                         }
130                         parametros.add("OK");
131                         parametros.add("ok");
132
133                         Post post = new Post();
134                         post.execute(parametros);
135                         String datos = post.get();
136
137                         if (datos.contains("nok"))
138                         {
139                                 Toast.makeText(getContext()
                                       , "Gracias por tu ayuda
                                       aunque este Restaurante
                                       ya est  registrado",
                                       Toast.LENGTH_LONG).show
                                       ();
140                         }
141
142                         else
143                         {
144                                 if (getWeb().length()>0)
```

```
145                                 {
146                                 if(datos.contains("ok") &&
                                       datos.contains("wok"))
147                                 {
148                                         Toast.makeText(
                                               getContext(), "
                                               Gracias por
                                               ense arnos un
                                               restaurante
                                               nuevo", Toast.
                                               LENGTH_LONG).
                                               show();
149                                 }
150                                 else
151                                 {
152                                         Toast.makeText(
                                               getContext(), "
                                               En estos
                                               momentos no se
                                               ha podido " + "
                                               guardar este
                                               restaurante,
                                               intentelo de
                                               nuevo m s tarde
                                               ", Toast.
                                               LENGTH_LONG).
                                               show();
153                                 }
154                                 }
155                                 else
156                                 {
157                                 if(datos.contains("ok"))
158                                 {
159                                         Toast.makeText(
                                               getContext(), "
                                               Gracias por
                                               ense arnos un
                                               restaurante
                                               nuevo", Toast.
                                               LENGTH_LONG).
                                               show();
160                                 }
161                                 else
```

```
162                                     {
163                                             Toast.makeText(
                                                   getContext(), "
                                                   En estos
                                                   momentos no se
                                                   ha podido " + "
                                                   guardar este
                                                   restaurante,
                                                   intentelo de
                                                   nuevo m s tarde
                                                   ", Toast.
                                                   LENGTH_LONG).
                                                   show();
164                                     }
165                                     }
166                             }
167
168                     }
169                     }
170                     catch (IOException e)
171                     {
172                             System.out.println(e.getMessage());
173                     }
174                     catch (InterruptedException e)
175                     {
176                             System.out.println(e.getMessage());
177                     }
178                     catch (ExecutionException e)
179                     {
180                             System.out.println(e.getMessage());
181                     }
182                     }
183             };
184
185             final OnClickListener eventoClear = new
                   OnClickListener()
186             {
187                     public void onClick(View v)
188                     {
189                             ERestaurante.setText("");
190                             EDireccion.setText("");
191                             EWeb.setText("");
192                     }
```

```
193             };
194
195             acept.setOnClickListener(eventoAcept);
196             clear.setOnClickListener(eventoClear);
197     }
198
199     /*****************************************************
200      * Obtiene el valor del string escrito en la etiqueta de
            restaurante
201      *
202      * @return String nombre del restaurante
203      *
204      *****************************************************/
205     public String getRestaurante()
206     {
207             return ERestaurante.getText().toString();
208     }
209
210     /*****************************************************
211      * Obtiene el valor del string escrito en la etiqueta de
            direccion
212      *
213      * @return String direccion del restaurante
214      *
215      *****************************************************/
216     public String getDireccion()
217     {
218             return EDireccion.getText().toString();
219     }
220
221     /*****************************************************
222      * Obtiene el valor del string escrito en la etiqueta de
            web
223      *
224      * @return String web del restaurante
225      *
226      *****************************************************/
227     public String getWeb()
228     {
229             return EWeb.getText().toString();
230     }
231     }
```

Clase ControlPuntuar.java

```
/*****************************************************
 *
 * Fichero: ControlPuntuar.java
 * Clase que modela la pantalla de la funcion puntuar de la
     aplicacion
 * @author Eduardo Campos de Diago
 * @version 1.2
 *
 *****************************************************/

package com.aplicacion.ui;

import java.util.ArrayList;
import java.util.concurrent.ExecutionException;

import com.aplicacion.conexion.Post;

import android.widget.Button;
import android.widget.Toast;
import android.widget.LinearLayout;
import android.widget.RadioGroup;
import android.content.Context;
import android.util.AttributeSet;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.EditText;


public class ControlPuntuar extends LinearLayout
{
 private EditText ERestaurante;
 private EditText EPlato;
 private RadioGroup grupo;
 private Button acept;
 private Button clear;

/*****************************************************
 * Constructor de la clase ControlPuntuar
 *
 * @param Context contesto de la aplicacion
 *
```

```
41   ****************************************************/
42   public ControlPuntuar(Context context)
43   {
44          super(context);
45          initialize();
46   }
47
48  /****************************************************
49   * Constructor de la clase ControlPuntuar
50   *
51   * @param Context contesto de la aplicacion
52   * @param AttributeSet Atributos
53   *
54   ****************************************************/
55   public ControlPuntuar(Context context, AttributeSet attrs)
56   {
57          super(context, attrs);
58          initialize();
59   }
60
61  /****************************************************
62   * Inicializa la interfaz para ser mostrada
63   *
64   ****************************************************/
65   private void initialize()
66   {
67          String interfaz = Context.LAYOUT_INFLATER_SERVICE;
68          LayoutInflater inflater = (LayoutInflater)
               getContext().getSystemService(interfaz);
69          inflater.inflate(R.layout.puntuar, this, true);
70
71          ERestaurante = (EditText)findViewById(R.id.
               edit_restaurante);
72          EPlato = (EditText)findViewById(R.id.edit_plato);
73          grupo = (RadioGroup) findViewById(R.id.grupo);
74          acept = (Button)findViewById(R.id.button_Puntuar);
75          clear = (Button)findViewById(R.id.button_Clear);
76
77          addEvents();
78   }
79
80
81  /****************************************************
```

```
82    * Anade los eventos a los botones de la pantalla
83    *
84    ************************************************/
85    private void addEvents()
86    {
87           final OnClickListener eventoAcept = new
                 OnClickListener()
88           {
89                  public void onClick(View v)
90                  {
91                  try
92                  {
93                  if (getRestaurante().length() < 1 ||
                       getPlato().length() < 1)
94                  {
95                         Toast.makeText(getContext(), "Debes
                             completar el nombre del
                            restaurante, " + "el del plato
                            que deseas puntuar y su
                            puntuaci n", Toast.LENGTH_LONG)
                            .show();
96                  }
97                  else
98                  {
99                         ArrayList<String> parametros = new
                            ArrayList<String>();
100
101                        parametros.add("option");
102                        parametros.add("2");
103                        parametros.add("nombre");
104                        parametros.add(getRestaurante().
                            toLowerCase());
105                        parametros.add("plato");
106                        parametros.add(getPlato().
                            toLowerCase());
107                        parametros.add("puntuacion");
108                        parametros.add(String.valueOf(
                            getPuntuacion()));
109                        parametros.add("OK");
110                        parametros.add("ok");
111
112                        Post post = new Post();
113                        post.execute(parametros);
```

```
114                         String datos = post.get();
115
116                         if(datos.contains("nok"))
117                         {
118                         if (datos.contains("nok0"))
119                         {
120                                 Toast.makeText(getContext()
                                    , "No tenemos ese
                                    restaurante guardado." +
                                     " Por favor, guardalo
                                    para poder puntuar el
                                    plato", Toast.
                                    LENGTH_LONG).show();
121                         }
122                         else
123                         {
124                                 Toast.makeText(getContext()
                                    , "En estos momentos no
                                    se ha podido " + "
                                    guardar tu puntuaci n,
                                    intentelo de nuevo  m  s
                                    tarde", Toast.
                                    LENGTH_LONG).show();
125                         }
126                         }
127                         else
128                         {
129                                 Toast.makeText(getContext()
                                    , "Gracias por puntuar
                                    este plato", Toast.
                                    LENGTH_LONG).show();
130                         }
131                 }
132                 }
133                 catch (InterruptedException e)
134                 {
135                         System.out.println(e.getMessage());
136                 }
137                 catch (ExecutionException e)
138                 {
139                         System.out.println(e.getMessage());
140                 }
141                 }
```

```
142         };
143
144         final OnClickListener eventoClear = new
               OnClickListener()
145         {
146                 public void onClick(View v)
147                 {
148                         ERestaurante.setText("");
149                         EPlato.setText("");
150                         grupo.clearCheck();
151                 }
152         };
153
154         acept.setOnClickListener(eventoAcept);
155         clear.setOnClickListener(eventoClear);
156  }
157
158
159 /*****************************************************
160  * Obtiene el valor del string escrito en la etiqueta de
       restaurante
161  *
162  * @return String nombre del restaurante
163  *
164  *****************************************************/
165  public String getRestaurante()
166  {
167         return ERestaurante.getText().toString();
168  }
169
170
171 /*****************************************************
172  * Obtiene el valor del string escrito en la etiqueta de
       plato
173  *
174  * @return String plato que se quiere puntuar
175  *
176  *****************************************************/
177  public String getPlato()
178  {
179         return EPlato.getText().toString();
180  }
181
```

```
182 /*****************************************************
183  * Obtiene el valor del valor de la puntuacion
184  *
185  * @return Int puntuacion dada
186  *
187  *****************************************************/
188  public int getPuntuacion()
189  {
190         int i = 0;
191         i = grupo.indexOfChild(findViewById(grupo.
                getCheckedRadioButtonId())) +1;
192         return i;
193  }
194 }
```

Clase Mapa.java

```
/*****************************************************
 *
 * Fichero: Mapa.java
 * Clase que modela un mapa
 * @author Eduardo Campos de Diago
 * @version 1.2
 *
 *****************************************************/

package com.aplicacion.ui;



import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.ExecutionException;

import serviOcios.Restaurantes;


import com.aplicacion.conexion.Post;
import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;
import com.google.android.maps.Overlay;
import com.google.gson.Gson;

import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.app.ProgressDialog;
import android.content.Context;
import android.util.Log;
import android.view.Menu;

public class Mapa extends MapActivity
```

```
42  {
43   private String comida;
44   private String dir;
45
46   private MapView mapa = null;
47   private MapController controlador = null;
48   private ProgressDialog pd;
49
50  /*****************************************************
51   * Es llamado cuando es creado
52   *
53   *
54   *****************************************************/
55   @Override
56   public void onCreate(Bundle savedInstanceState)
57   {
58          super.onCreate(savedInstanceState);
59          setContentView(R.layout.layout_mapa);
60
61          //Obtenemos una referencia al control MapView
62          mapa = (MapView)findViewById(R.id.mapa);
63
64          //Mostramos los controles de zoom sobre el mapa
65          mapa.setBuiltInZoomControls(true);
66
67          controlador = mapa.getController();
68
69          Bundle bundle = getIntent().getExtras();
70          comida =  bundle.getString("comida");
71          dir = bundle.getString("dir");
72
73          if (dir.length()!=0)
74          {
75                  try
76                  {
77                          Geocoder geocoder = new Geocoder(
                               this);
78
79                          List<Address> addresses;
80
81                          addresses = geocoder.
                               getFromLocationName(dir, 1);
```

```
82                          pd = ProgressDialog.show(this, "
                               Localizaci n", "Esperando
                               obtener datos...");
83
84                          if(addresses.size() > 0)
85                          {
86                              double latitude = addresses.get
                                   (0).getLatitude();
87                              double longitude = addresses.
                                   get(0).getLongitude();
88
89                              Location location = new
                                   Location(LocationManager.
                                   GPS_PROVIDER);
90                              location.setLatitude(latitude);
91                              location.setLongitude(longitude
                                   );
92                               refreshMap(location);
93                          }
94                  }
95                  catch (IOException e)
96                  {
97                          System.out.println(e.getMessage());
98                  }
99          }
100         else
101         {
102                 configGPS();
103                 pd = ProgressDialog.show(this, "
                       Localizaci n", "Esperando localizaci n
                        GPS...");
104         }
105  }
106
107 /*****************************************************
108  * Es llamado cuando es creado para mostrar las opciones
        del menu
109  *
110  *
111  *****************************************************/
112  @Override
113  public boolean onCreateOptionsMenu(Menu menu)
114  {
```

```
115         getMenuInflater().inflate(R.menu.activity_main,
               menu);
116         return true;
117  }
118
119  @Override
120  protected boolean isRouteDisplayed()
121  {
122         // TODO Auto-generated method stub
123         return false;
124  }
125
126 /*****************************************************
127  * Obtiene los datos del GPS del dispositivo
128  *
129  *
130  *****************************************************/
131  private void configGPS()
132  {
133         LocationManager mLocationManager;
134         LocationListener mLocationListener;
135
136         mLocationManager = (LocationManager)
               getSystemService(Context.LOCATION_SERVICE);
137
138         mLocationListener = new MyLocationListener();
139
140         mLocationManager.requestLocationUpdates(
               LocationManager.GPS_PROVIDER, 5000, 10,
               mLocationListener);
141  }
142
143 /*****************************************************
144  * Actualiza el mapa acorde a la posicion que se le pase
145  *
146  * @param Location location Localizacion que se quiere
        mostrar
147  *
148  *****************************************************/
149  private void refreshMap(Location location)
150  {
151         if (location != null)
152         {
```

```
153                     pd.dismiss();
154                     final List<Overlay> overlays = mapa.
                           getOverlays();
155 
156                     overlays.clear();
157                     Restaurantes restaurantes = null;
158 
159                     GeoPoint point = new GeoPoint( (int) (
                           location.getLatitude() * 1000000), (int)
                            (location.getLongitude() * 1000000));
160                     controlador.setZoom(18);
161                     controlador.animateTo(point);
162 
163                     MapOverlay myMapOverlay = new MapOverlay(
                           getResources().getDrawable(R.drawable.
                           marcador), "Usted", point);
164                     overlays.add(myMapOverlay);
165 
166                     restaurantes = obtenerRestaurantes(location
                           );
167 
168                     myMapOverlay = new MapOverlay(getResources
                           ().getDrawable(R.drawable.icono_rest),
                           restaurantes);
169                     overlays.add(myMapOverlay);
170             }
171 
172     }
173 
174     /***************************************************
175      * Obtiene Restaurantes que hay en torno a una posicion
176      *
177      * @param Location location Localizacion en torno a la que
                buscar
178      *
179      * @retunr Restaurantes Objeto de la clase Restaurantes con
                los
180      * resultados obtenidos
181      *
182      ***************************************************/
183     @SuppressWarnings("unchecked")
184     private Restaurantes obtenerRestaurantes(Location location
           )
```

```
185   {
186           Restaurantes restaurantes = null;
187           ArrayList<String> parametros = new ArrayList<String
                  >();
188
189           parametros.add("option");
190           parametros.add("0");
191           parametros.add("X");
192           parametros.add(String.valueOf(location.getLongitude
                  ()));
193           parametros.add("Y");
194           parametros.add(String.valueOf(location.getLatitude
                  ()));
195           parametros.add("comida");
196           parametros.add(comida);
197           parametros.add("OK");
198           parametros.add("ok");
199
200           Post post = new Post();
201           post.execute(parametros);
202
203           try
204           {
205                   String datos = post.get();
206                   Log.e("log_tag", datos);
207                   Gson gson = new Gson();
208
209                   restaurantes = gson.fromJson(datos,
                          Restaurantes.class);
210           }
211           catch (InterruptedException e)
212           {
213                   Log.e("log_tag", e.getMessage());
214           }
215           catch (ExecutionException e)
216           {
217                   Log.e("log_tag", e.getMessage());
218           }
219           return restaurantes;
220   }
221
222  /**************************************************
223   *
```

```
224  * Fichero: MyLocationListener.java
225  * Clase que escucha los eventos del mapa
226  * @author Eduardo Campos de Diago
227  * @version 1.2
228  *
229  ************************************************/
230  public class MyLocationListener implements
        LocationListener
231  {
232          double longitud;
233          double latitud;
234          public void onLocationChanged(Location location)
235          {
236                  longitud = location.getLongitude();
237                  latitud = location.getLatitude();
238
239                  Log.d("activityMapa", "Longitud: " + String
                        .valueOf(longitud) );
240                  Log.d("activityMapa", "Latitud: " + String.
                        valueOf(latitud) );
241
242                  refreshMap(location);
243          }
244
245
246          public void onProviderDisabled(String location) {
247                  // TODO Auto-generated method stub
248          }
249
250          public void onProviderEnabled(String provider) {
251                  // TODO Auto-generated method stub
252          }
253
254          public void onStatusChanged(String provider, int
                status, Bundle extras) {
255                  // TODO Auto-generated method stub
256          }
257   }
258  }
```

Clase MapOverlay.java

```
/*****************************************************
 *
 * Fichero: MapOverlay.java
 * Clase que gestiona el overlay de un mapa de google
 * @author Eduardo Campos de Diago
 * @version 1.2
 *
 *****************************************************/

package com.aplicacion.ui;

import java.util.Iterator;

import serviOcios.Restaurante;
import serviOcios.Restaurantes;

import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.DialogInterface.OnClickListener;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.Point;
import android.graphics.drawable.Drawable;

import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapView;
import com.google.android.maps.Overlay;
import com.google.android.maps.Projection;

public class MapOverlay extends Overlay
{
 private Drawable imagen;
 private int xOffset;
 private int yOffset;
 private String text = "";
 private Restaurantes restaurantes = new Restaurantes();
 private GeoPoint punto;

/*****************************************************
 * Constructor de la clase MapOverlay
 *
```

```
42   * @param Drawable draw
43   * @param String text Texto que se desea excribir
44   * @param GeoPoint point Punto que se desea representar
45   *
46   ****************************************************/
47   public MapOverlay (Drawable draw, String text, GeoPoint
        point)
48   {
49          this.imagen = draw;
50                  this.text = text;
51                  this.punto = point;
52   }
53
54  /****************************************************
55   * Constructor de la clase MapOverlay
56   *
57   * @param Drawable draw
58   * @param Restaurantes r Restaurantes que se quieren
        mostrar
59   *
60   ****************************************************/
61   public MapOverlay (Drawable draw, Restaurantes r)
62   {
63          this.imagen = draw;
64          this.restaurantes = r;
65   }
66
67  /****************************************************
68   * Dibuja el overlay
69   *
70   * @param Canvas canvas
71   * @param MapView mapView
72   * @param boolean shadow
73   *
74   ****************************************************/
75   public void draw (Canvas canvas, MapView mapView, boolean
        shadow)
76   {
77          if (!shadow)
78          {
79                  final int intrinsicWidth = imagen.
                        getIntrinsicWidth();
```

```
80                  final int intrinsicHeight = imagen.
                       getIntrinsicHeight();
81                  imagen.setBounds(0, 0, intrinsicWidth,
                       intrinsicHeight);
82
83                  xOffset = -(intrinsicWidth / 2);
84                  yOffset = -(intrinsicHeight / 2);
85
86                  Paint paint = new Paint();
87                  paint.setARGB(250, 0, 0, 0);
88                  paint.setAntiAlias(true);
89                  paint.setFakeBoldText(true);
90
91                  if (text != "")
92                  {
93                          Point point2 = new Point();
94                          Projection p = mapView.
                             getProjection();
95                          p.toPixels(punto, point2);
96
97                          canvas.drawText(text, point2.x-
                             intrinsicWidth, point2.y+
                             intrinsicHeight, paint);
98                          super.draw(canvas, mapView, shadow)
                             ;
99                          drawAt(canvas, imagen, point2.x +
                             xOffset, point2.y + yOffset,
                             shadow);
100                 }
101
102                 Iterator<Restaurante> ite = restaurantes.
                       getRestaurantes().iterator();
103                 while(ite.hasNext())
104                 {
105                         Restaurante rest = ite.next();
106                         punto = new GeoPoint( (int) (rest.
                             getCoordY() * 1000000), (int) (
                             rest.getCoordX() * 1000000));
107                         Point point2 = new Point();
108                         Projection p = mapView.
                             getProjection();
109                         p.toPixels(punto, point2);
110
```

```
111                         canvas.drawText(rest.getNombre(),
                               point2.x-intrinsicWidth, point2.
                               y+intrinsicHeight, paint);
112                         super.draw(canvas, mapView, shadow)
                               ;
113                         drawAt(canvas, imagen, point2.x +
                               xOffset, point2.y + yOffset,
                               shadow);
114                 }
115         }
116 }
117
118 @Override
119 public boolean onTap(GeoPoint point, MapView mapView)
120 {
121         double margen = 0.0001;
122         double lat, lon;
123
124         lat = point.getLatitudeE6()/1E6;
125         lon = point.getLongitudeE6()/1E6;
126
127         Iterator<Restaurante> ite = restaurantes.
               getRestaurantes().iterator();
128         while(ite.hasNext())
129         {
130                 Restaurante restaurante = ite.next();
131                 if (getDistance(lat, lon, restaurante.
                       getCoordY(), restaurante.getCoordX()) <
                       margen)
132                 {
133                         AlertDialog.Builder builder = new
                               AlertDialog.Builder(mapView.
                               getContext());
134
135                         builder.setTitle("Informacion de
                               restaurante");
136                         int i = (int) (restaurante.
                               getPuntuacion()/1);
137                         switch (i)
138                         {
139                                 case -1:
140                                         mensaje = "Nombre:
                                               "+restaurante.
```

```
                                        getNombre()+
141                                                 "\
                                                    nDirecci n
                                                    :
                                                    "+
                                                    restaurante
                                                    .
                                                    getDireccion
                                                    ()
                                                    +
142                                                 "\
                                                    nPuntuaci n
                                                    :
                                                    Este
                                                    restaurante
                                                    a n
                                                    no
                                                    ha
                                                    sido
                                                    puntuado
                                                    ";
143                             break;
144                             case 0:
145                                     mensaje = "Nombre:
                                        "+restaurante.
                                        getNombre()+
146                                     "\nDirecci n: "+
                                        restaurante.
                                        getDireccion()+
147                                     "\nPuntuaci n:
                                        Malo (" +
                                        restaurante.
                                        getPuntuacion()
```

```
                                                    +")";
148                                 break;
149                                 case 1:
150                                         mensaje = "Nombre:
                                                    "+restaurante.
                                                    getNombre()+
151                                         "\nDirecci n: "+
                                                    restaurante.
                                                    getDireccion()+
152                                         "\nPuntuaci n:
                                                    Regular (" +
                                                    restaurante.
                                                    getPuntuacion()
                                                    +")";
153                                 break;
154                                 case 2:
155                                         mensaje = "Nombre:
                                                    "+restaurante.
                                                    getNombre()+
156                                         "\nDirecci n: "+
                                                    restaurante.
                                                    getDireccion()+
157                                         "\nPuntuaci n:
                                                    Bueno (" +
                                                    restaurante.
                                                    getPuntuacion()
                                                    +")";
158                                 break;
159                                 case 3: case 4:
160                                         mensaje = "Nombre:
                                                    "+restaurante.
                                                    getNombre()+
161                                         "\nDirecci n: "+
                                                    restaurante.
                                                    getDireccion()+
162                                         "\nPuntuaci n: Muy
                                                     Bueno (" +
                                                    restaurante.
                                                    getPuntuacion()
                                                    +")";
163                                 break;
164                         }
165                         builder.setMessage(mensaje);
```

```
166                             builder.setPositiveButton("Aceptar
                                   ", new OnClickListener()
167                             {
168                                     public void onClick(
                                           DialogInterface dialog,
                                           int which)
169                                     {
170                                             dialog.cancel();
171                                     }
172                             });
173                             builder.create();
174                             builder.show();
175                     }
176             }
177             return true;
178     }
179
180     /*****************************************************
181      * Obtiene la distancia entre dos puntos
182      *
183      * @param lat_a double latitud del punto a
184      * @param lon_a double longitud del punto a
185      * @param lat_b double latitud del punto b
186      * @param lon_b double longitud del punto b
187      * @return double con la distacina entre los dos puntos
188      *
189      *****************************************************/
190     private double getDistance(double lat_a,double lon_a,
           double lat_b, double lon_b)
191     {
192             double dLat = lat_b-lat_a;
193             double dLon = lon_b-lon_a;
194             double a = dLat*dLat;
195             double b = dLon*dLon;
196             double c = Math.sqrt(a+b);
197             return (c);
198     }
199     }
```

B.1.2. Paquete com.aplicacion.conexion

Clase Post.java

```
1  /*****************************************************
2   *
3   * Fichero: Post.java
4   * Clase Encargada de realizar la conexion HTTP mediante
       mensajes POST entre
5   * el cliente y el servidor
6   * @author Eduardo Campos de Diago
7   * @version 1.2
8   *
9   *****************************************************/
10
11 package com.aplicacion.conexion;
12
13 import java.io.BufferedReader;
14 import java.io.InputStream;
15 import java.io.InputStreamReader;
16 import java.util.ArrayList;
17
18 import org.apache.http.HttpEntity;
19 import org.apache.http.HttpResponse;
20 import org.apache.http.client.HttpClient;
21 import org.apache.http.client.entity.UrlEncodedFormEntity;
22 import org.apache.http.client.methods.HttpPost;
23 import org.apache.http.impl.client.DefaultHttpClient;
24 import org.apache.http.message.BasicNameValuePair;
25
26
27 import android.os.AsyncTask;
28 import android.util.Log;
29
30 public class Post extends AsyncTask<ArrayList<String>,
       String, String>
31 {
32  private final String server = "http://eiffel.itba.edu.ar/
       resto/";
33
34  private InputStream is = null;
35  private String respuesta = "";
36
```

```
37  /*****************************************************
38   * Envia un mensaje post con los parametros deseados a la
        direccion
39   * indicada
40   *
41   * @param ArrayList parametros Parametros que se quieren
        enviar
42   * @param String URL URL del servicio
43   *
44   *****************************************************/
45   private void conectaPost(ArrayList parametros, String URL)
46   {
47           ArrayList nameValuePairs;
48           try
49           {
50                   HttpClient httpclient = new
                        DefaultHttpClient();
51                   HttpPost httppost = new HttpPost(URL);
52                   nameValuePairs = new ArrayList();
53
54                   if (parametros != null)
55                   {
56                           for (int i = 0; i < parametros.size
                                () - 1; i += 2)
57                           {
58                                   nameValuePairs.add(new
                                        BasicNameValuePair((
                                        String)parametros.get(i)
                                        ,(String)parametros.get(
                                        i + 1)));
59                           }
60                           httppost.setEntity(new
                                UrlEncodedFormEntity(
                                nameValuePairs));
61                   }
62
63                   HttpResponse response = httpclient.execute(
                        httppost);
64                   HttpEntity entity = response.getEntity();
65                   is = entity.getContent();
66           }
67           catch (Exception e)
68           {
```

```
69                  Log.e("log_tag", "Error in http connection
                        " + e.toString());
70          }
71          finally
72          {
73
74          }
75   }
76
77  /************************************************
78   * Obtiene una respuesta del servidor correspondiente a la
        peticion
79   * realizada
80   *
81   * @return String (codigo completo de la respuesta HTTP)
82   *
83   ************************************************/
84   private String getRespuestaPost()
85   {
86          try
87          {
88                  BufferedReader reader = new BufferedReader(
                        new InputStreamReader(is, "iso-8859-1"),
                            8);
89                  StringBuilder sb = new StringBuilder();
90                  String line = null;
91                  while ((line = reader.readLine()) != null)
92                  {
93                          sb.append(line + "\n");
94                  }
95                  is.close();
96                  respuesta = sb.toString();
97                  //Log.e("log_tag", "Cadena JSon " +
                        respuesta);
98          }
99          catch (Exception e)
100         {
101                 //Log.e("log_tag", "Error converting result
                        " + e.toString());
102         }
103         return respuesta;
104  }
105
```

```
106 /*****************************************************
107  * Obtiene un JSon con la informacion pedida por el cliente
108  *
109  * @param ArrayList parametros Parametros que se quieren
         enviar
110  * @return String Json con la informacion de la respuesta
111  *
112  *****************************************************/
113  public String getServerDataString(ArrayList<String>
         parametros)
114  {
115         conectaPost(parametros, server);
116         if (is != null)
117         {
118                 getRespuestaPost();
119         }
120         if (respuesta != null && respuesta.trim() != "")
121         {
122                 respuesta = respuesta.substring(respuesta.
                    indexOf("{"), respuesta.lastIndexOf("}")
                    +1);
123                 return respuesta;
124         }
125         else
126         {
127                 return null;
128         }
129  }
130
131  @Override
132  protected String doInBackground(ArrayList<String>...
         params)
133  {
134         return getServerDataString(params[0]);
135  }
136 }
```

B.1.3. Paquete serviOcios

Clase Restaurante.java

```
1  /*****************************************************
2   *
3   * Fichero: Restaurante.java
4   * Clase que caracteriza un restaurante
5   * @author Eduardo Campos de Diago
6   * @version 1.2
7   *
8   *****************************************************/
9
10 package serviOcios;
11
12 import java.util.ArrayList;
13
14
15 public class Restaurante
16 {
17  public int id;
18  public String nombre;
19  public String direccion;
20  public double coord_X;
21  public double coord_Y;
22  public float puntuacion;
23  public ArrayList<String> web;
24  public ArrayList<String> platos;
25
26 /*****************************************************
27  * Constructor de la clase Restaurante
28  *
29  * @param int i indice de la bbdd
30  * @param String n nombre
31  * @param String d direcci n
32  * @param double x coordenada x
33  * @param double y coordenada y
34  * @param float p puntuaci n
35  *
36  *****************************************************/
37  public Restaurante(int i, String n, String d, double x,
       double y, float p)
38  {
```

```
39          this.id = i;
40          this.nombre = n;
41          this.direccion = d;
42          this.coord_X = x;
43          this.coord_Y = y;
44          this.puntuacion = p;
45          this.web = new ArrayList<String>();
46          this.platos = new ArrayList<String>();
47   }
48
49   /*****************************************************
50   * Constructor vacio de la clase restaurante
51   *
52   *****************************************************/
53   public Restaurante()
54   {
55          this.id = -1;
56          this.nombre = "";
57          this.direccion = "";
58          this.coord_X = -1;
59          this.coord_Y = -1;
60          this.puntuacion = -1;
61          this.web = new ArrayList<String>();
62          this.platos = new ArrayList<String>();
63   }
64
65  /*****************************************************
66   * Constructor de la clase restaurante
67   *
68   * @param int i indice de la bbdd
69   * @param String n nombre
70   * @param String d direcci n
71   * @param double x coordenada x
72   * @param double y coordenada y
73   * @param float p puntuaci n
74   * @param ArrayList<String> w lista de webs
75   * @param ArrayLis<String> pl lista de platos
76   *
77   *****************************************************/
78   public Restaurante(int i, String n, String d, double x,
        double y, float p, ArrayList<String> w, ArrayList<
        String> pl)
79   {
```

```
80          this.id = i;
81          this.nombre = n;
82          this.direccion = d;
83          this.coord_X = x;
84          this.coord_Y = y;
85          this.puntuacion = p;
86          this.web = w;
87          this.platos = pl;
88   }
89
90  /*****************************************************
91   * Obtiene el id de un restaurante
92   *
93   * @return int
94   *
95   *****************************************************/
96   public int getId()
97   {
98          return (this.id);
99   }
100
101 /*****************************************************
102  * Modifica el id de un restaurante
103  *
104  * @param int i nuevo id
105  *
106  *****************************************************/
107  public void setId(int i)
108  {
109         this.id = i;
110  }
111
112 /*****************************************************
113  * Obtiene el nombre de un restaurante
114  *
115  * @return String
116  *
117  *****************************************************/
118  public String getNombre()
119  {
120         return (this.nombre);
121  }
122
```

```
123  /*****************************************************
124   * Modifica el nombre de un restaurante
125   *
126   * @param String n nuevo nombre
127   *
128   *****************************************************/
129   public void setNombre(String n)
130   {
131          this.nombre = n;
132   }
133
134  /*****************************************************
135   * Obtiene la direccion de un restaurante
136   *
137   * @return String
138   *
139   *****************************************************/
140   public String getDireccion()
141   {
142          return (this.direccion);
143   }
144
145  /*****************************************************
146   * Modifica la direccion de un restaurante
147   *
148   * @param String d nueva direcci n
149   *
150   *****************************************************/
151   public void setDireccion(String d)
152   {
153          this.direccion = d;
154   }
155
156  /*****************************************************
157   * Obtiene la coordenada x de un restaurante
158   *
159   * @return double
160   *
161   *****************************************************/
162   public double getCoordX()
163   {
164          return (this.coord_X);
165   }
```

```
166
167  /*****************************************************
168   * Modifica la coordenada x de un restaurante
169   *
170   * @param double x nueva coordenada x
171   *
172   *****************************************************/
173   public void setCoordX(double x)
174   {
175          this.coord_X = x;
176   }
177
178  /*****************************************************
179   * Obtiene la coordenada y de un restaurante
180   *
181   * @return double
182   *
183   *****************************************************/
184   public double getCoordY()
185   {
186          return (this.coord_Y);
187   }
188
189  /*****************************************************
190   * Modifica la coordenada y de un restaurante
191   *
192   * @param double y nueva coordenada y
193   *
194   *****************************************************/
195   public void setCoordY(double y)
196   {
197          this.coord_Y = y;
198   }
199
200  /*****************************************************
201   * Obtiene la puntuacion de un restaurante
202   *
203   * @return float
204   *
205   *****************************************************/
206   public float getPuntuacion()
207   {
208          return (this.puntuacion);
```

```
209   }
210
211  /**************************************************
212   * Modifica la puntuacion de un restaurante
213   *
214   * @param float p nueva puntuacion
215   *
216   **************************************************/
217   public void setPuntuacion(float p)
218   {
219          this.puntuacion = p;
220   }
221
222  /**************************************************
223   * Obtiene la lista de paginas web de un restaurante
224   *
225   * @return ArrayList<String>
226   *
227   **************************************************/
228   public ArrayList<String> getWeb()
229   {
230          return (this.web);
231   }
232
233  /**************************************************
234   * Modifica la lista de paginas web de un restaurante
235   *
236   * @param ArrayList<String> w nueva lista de paginas web
237   *
238   **************************************************/
239   public void setWeb(ArrayList<String> w)
240   {
241          this.web = w;
242   }
243
244  /**************************************************
245   * Anade una web a la lista de paginas web de un
        restaurante
246   *
247   * @param String w nueva paginas web
248   *
249   **************************************************/
250   public void addWeb(String w)
```

```
251   {
252           this.web.add(w);
253   }
254
255  /*****************************************************
256   * Obtiene la lista de platos de un restaurante
257   *
258   * @return ArrayList<String>
259   *
260   *****************************************************/
261   public ArrayList<String> getPlatos()
262   {
263           return (this.platos);
264   }
265
266  /*****************************************************
267   * Modifica la lista de platos de un restaurante
268   *
269   * @param ArrayList<String> p nueva lista de platos
270   *
271   *****************************************************/
272   public void setPlatos(ArrayList<String> p)
273   {
274           this.platos = p;
275   }
276  }
```

Clase Restaurantes.java

```
/*****************************************************
 *
 * Fichero: Restaurantes.java
 * Clase que caracteriza una lista de restaurante
 * @author Eduardo Campos de Diago
 * @version 1.2
 *
 *****************************************************/

package serviOcios;

import java.util.ArrayList;

public class Restaurantes
{
 public ArrayList<Restaurante> restaurantes;

/*****************************************************
 * Constructor vacio de la clase restaurantes
 *
 *****************************************************/
 public Restaurantes()
 {
        this.restaurantes = new ArrayList<Restaurante>();
 }

/*****************************************************
 * Constructor de la clase Restaurante
 *
 * @param ArrayList<String> r Lista de restaurantes
 *
 *****************************************************/
 public Restaurantes(ArrayList<Restaurante> r)
 {
        this.restaurantes = r;
 }

/*****************************************************
 * Obtiene la lista de restaurantes
 *
 * @return ArrayList<String
```

```
42   *
43   ****************************************************/
44   public ArrayList<Restaurante> getRestaurantes()
45   {
46          return restaurantes;
47   }
48
49  /****************************************************
50   * Modifica la lista de restaurantes
51   *
52   * @return ArrayList<String> r Lista de restaurantes
53   *
54   ****************************************************/
55   public void setRestaurantes(ArrayList<Restaurante> r)
56   {
57          this.restaurantes = r;
58   }
59
60  /****************************************************
61   * A ade un restaurante a la lista de restaurantes
62   *
63   * @return Restaurante r restaurante que se quiere anadir a
         la lista
64   *
65   ****************************************************/
66   public void add(Restaurante r)
67   {
68          this.restaurantes.add(r);
69   }
70
71  /****************************************************
72   * Borra un restaurante a la lista de restaurantes
73   *
74   * @param Restaurante r restaurante que se quiere borrar a
         la lista
75   *
76   ****************************************************/
77   public void remove(Restaurante r)
78   {
79          if (restaurantes.contains(r))
80          {
81                  for(int i = 0; i<restaurantes.size(); i++)
82                  {
```

```
83                                  if (restaurantes.get(i).equals(r))
84                                  {
85                                          this.restaurantes.remove(i)
                                               ;
86                                  }
87                          }
88                  }
89      }
90  }
```

B.1.4. Paquete core

Clase Gestor.java

```
/*****************************************************
 *
 * Fichero: Gestor.java
 * Clase encargada de realizar la conexion a la BBDD y de
    responder las
 * peticiones de los clientes (aplicacion)
 * @author Eduardo Campos de Diago
 * @version 1.2
 *
 *****************************************************/

package core;

import java.io.BufferedReader;
import serviOcios.*;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLConnection;
import java.sql.*;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Scanner;

import com.google.gson.Gson;


public class Gestor
{
 //private static String dominio = "eiffel.itba.edu.ar";
 //private static String sid = "ITBA";
 private static String user = "camposdediago";
 private static String pass = "gi53pw21";
 static Connection cn;

 final static String unDriver = "oracle.jdbc.driver.
    OracleDriver";
```

```
37  final static String URLConn = "jdbc:oracle:thin:gi53pw21/
       camposdediago@eiffel.itba.edu.ar:1521:ITBA";
38
39  private static Gson gson = new Gson();
40
41  static Scanner teclado;
42
43  public static void main (String[] args)
44  {
45          String resultado = "";
46          int opcion;
47          String aux;
48          teclado = new Scanner(System.in);
49
50          opcion = menu();
51          while (opcion != 9)
52          {
53          switch (opcion)
54          {
55                  case 0: //Consulta insertando coordenadas
56                          double x;
57                          double y;
58                          System.out.println("\nInserte
                               coordenada X:");
59                          aux = teclado.next();
60                          x = Double.parseDouble(aux);
61                          System.out.println("\nInserte
                               coordenada Y:");
62                          aux = teclado.next();
63                          y = Double.parseDouble(aux);
64                          System.out.println("\nInserte
                               comida:");
65                          aux = teclado.next();
66                          System.out.println("x: "+x+"\ny:"+y
                               +"\ncomida:"+aux);
67
68                          resultado = obtenerRestaurantes(x,
                               y, aux);
69                          System.out.println(resultado);
70
71                          break;
72                  case 1:
73                          String direccion = "";
```

```
74                          String nombre = "";
75                          String web = "";
76
77                          System.out.println("\nInserte
                               nombre:");
78                          nombre = teclado.next();
79                          System.out.println("\nInserte
                               direcci n:");
80                          direccion = teclado.next();
81                          System.out.println("\nInserte
                               coordenada X:");
82                          aux = teclado.next();
83                          x = Double.parseDouble(aux);
84                          System.out.println("\nInserte
                               coordenada Y:");
85                          aux = teclado.next();
86                          y = Double.parseDouble(aux);
87                          System.out.println("\nInserte web
                               :");
88                          web = teclado.next();
89
90
91                          resultado = insertRestaurante(
                               nombre, direccion, x, y, web);
92                          System.out.println(resultado);
93
94                          break;
95                  case 2: //puntuar un plato
96                          float p;
97                          System.out.println("\nInserte rest
                               :");
98                          nombre = teclado.next();
99                          System.out.println("\nInserte plato
                               :");
100                         direccion = teclado.next();
101                         System.out.println("\nInserte
                               puntuacion:");
102                         aux = teclado.next();
103                         p = Float.parseFloat(aux);
104
105                         resultado = insertPlato(nombre,
                               direccion, p);
106                         System.out.println(resultado);
```

```
107
108                         break;
109                 case 9:
110                         System.out.println("\nSaliendo de
                               la aplicacion. Hasta luego.");
111                         break;
112                 default:
113                         System.out.println("\nOpcion no
                               valida. ");
114                         break;
115         }
116         opcion = menu();
117         }
118
119  }
120
121  private static int menu()
122  {
123         int opcion = -1;
124
125         System.out.print("\nOPERACIONES CON LA BBDD\n");
126         System.out.print("\n\nElija su opcion:\n\t0. " + "
               Consultar mediante coordenadas\n\t1. " + "
               Insertar un restaurante\n\t2. " + "Puntuar un
               plato\n" + "\n9.- Salir\n");
127         opcion = teclado.nextInt();
128
129         return (opcion);
130  }
131
132 /*****************************************************
133  * Funcion para conectarse a la bbdd
134  *
135  *****************************************************/
136  private static void conectarBBDD() throws
        ClassNotFoundException, SQLException,
        InstantiationException, IllegalAccessException
137  {
138         System.out.println("Conectando a la BBDD...");
139         Class.forName(unDriver);
140         cn = DriverManager.getConnection(URLConn, user,
               pass);
141         System.out.println("Conectado!!!");
```

```
142   }
143
144  /**************************************************
145   * Obtiene restaurantes dadas unas coordenadas y una comida
146   *
147   * @param double x coordenada x (latitud)
148   * @param double y coordenada y (longitud)
149   * @param String comida comida que se desea buscar
150   * @return String json con la respuesta obtenida
151   *
152   **************************************************/
153   public static String obtenerRestaurantes(double x, double
          y, String comida)
154   {
155           String resJson = "";
156           ArrayList<Restaurantes> resultados = new ArrayList<
              Restaurantes>();
157           Restaurantes restaurantes = new Restaurantes();
158           Restaurantes rests = new Restaurantes();
159
160           restaurantes = restaurantesEntornoADir(x, y);
161           resultados = restaurantesComidaBBDD(restaurantes,
              comida, rests);
162           resultados = restaurantesComidaWeb(resultados.get
              (0), comida, resultados.get(1));
163
164           resJson = gson.toJson(resultados.get(1));
165           return resJson;
166   }
167
168  /**************************************************
169   * Obtiene restaurantes que tienen una comida deseada tras
          buscar
170   * en la pagina web de dichos restaurantes
171   *
172   * @param Restaurantes restaurantes lista de restaurantes
          que se quiere comprobar si tienen la comida deseada
173   * @param String comida comida que se desea buscar
174   * @param Restaurantes rests lisya der estaurantes que ya
          se sabe que tienen la comida buscada.
175   * @return ArrayList<Restaurantes>
176   *
177   **************************************************/
```

```
178   private static ArrayList<Restaurantes>
          restaurantesComidaWeb(Restaurantes restaurantes, String
           comida, Restaurantes rests)
179   {
180           ResultSet resultSetWeb = null;
181           String query;
182           Statement statement;
183           ArrayList<Restaurantes> resTurn = new ArrayList<
                 Restaurantes>();
184           boolean esta=false;
185           Restaurantes aux = new Restaurantes();
186
187
188           try
189           {
190                   statement = cn.createStatement();
191
192                   Iterator<Restaurante> ite = restaurantes.
                         getRestaurantes().iterator();
193                   while(ite.hasNext())
194                   {
195                           ArrayList<String> plas = new
                                 ArrayList<String>();
196                           esta=false;
197                           Restaurante rest = new Restaurante
                                 ();
198                           rest = (Restaurante) ite.next();
199                           query = "SELECT * FROM web WHERE
                                 id_restaurante = "+rest.getId();
200
201                           resultSetWeb = statement.
                                 executeQuery(query);
202
203                           while (resultSetWeb.next())
204                           {
205                                   String contenido = "";
206                                   try
207                           {
208                                   URL url;
209                                   url = new URL(resultSetWeb.
                                         getString("uri"));
210
```

```
211                                 URLConnection uc = url.
                                       openConnection();
212                                 uc.connect();
213
214                                 InputStreamReader is = new
                                       InputStreamReader(uc.
                                       getInputStream(), "ISO
                                       -8859-1");
215                                 System.out.println(is.
                                       getEncoding());
216                                 BufferedReader in = new
                                       BufferedReader(is);
217
218                                 String inputLine;
219
220                                 while ((inputLine = in.
                                       readLine()) != null)
221                                 {
222                                         contenido +=
                                             inputLine + "\n
                                             ";
223                                 }
224                                 in.close();
225                                 contenido = contenido.
                                       toLowerCase();
226                         }
227                                 catch (
                                       MalformedURLException e)
228                                 {
229                                         System.out.println(
                                             e.getMessage());
230                                 }
231                                 catch (IOException e)
232                                 {
233                                         System.out.println(
                                             e.getMessage());
234                                 }
235                                 if (contenido.contains(
                                       comida.toLowerCase()))
236                                         esta = true;
237                         }
238                         if (esta)
239                         {
```

```
240                                     rest.setWeb(plas);
241                                     rests.add(rest);
242                             }
243                             else
244                             {
245                                     aux.add(rest);
246                             }
247
248                     }
249             }
250             catch (SQLException e)
251             {
252                     System.out.println(e.getMessage());
253             }
254
255             resTurn.add(aux);
256             resTurn.add(rests);
257             return resTurn;
258     }
259
260     /**************************************************
261      * Obtiene restaurantes que tienen una comida deseada tras
           buscar
262      * en la bbdd
263      *
264      * @param Restaurantes restaurantes lista de restaurantes
           que se quiere comprobar si tienen la comida deseada
265      * @param String comida comida que se desea buscar
266      * @param Restaurantes rests lisya der estaurantes que ya
           se sabe que tienen la comida buscada.
267      * @return ArrayList<Restaurantes>
268      *
269      **************************************************/
270     private static ArrayList<Restaurantes>
           restaurantesComidaBBDD(Restaurantes restaurantes,
           String comida, Restaurantes rests)
271     {
272             ResultSet resultSetPlat = null;
273             String query;
274             Statement statement;
275             ArrayList<Restaurantes> resTurn = new ArrayList<
                   Restaurantes>();
276             boolean esta=false;
```

```
277         Restaurantes aux = new Restaurantes();
278
279
280         try
281         {
282                 statement = cn.createStatement();
283
284                 Iterator<Restaurante> ite = restaurantes.
                       getRestaurantes().iterator();
285                 while(ite.hasNext())
286                 {
287                         ArrayList<String> plas = new
                           ArrayList<String>();
288                         esta=false;
289                         Restaurante rest = new Restaurante
                           ();
290                         rest = (Restaurante) ite.next();
291                         query = "SELECT * FROM plato WHERE
                           id_restaurante = "+rest.getId()
                           +" AND 1 = instr(plato, '"+
                           comida+"')";
292                         resultSetPlat = statement.
                           executeQuery(query);
293
294                         while (resultSetPlat.next())
295                         {
296                                 plas.add(resultSetPlat.
                                   getString("plato"));
297                                 esta = true;
298                         }
299                         if (esta)
300                         {
301                                 rest.setPlatos(plas);
302                                 rests.add(rest);
303                         }
304                         else
305                         {
306                                 aux.add(rest);
307                         }
308                 }
309         }
310         catch (SQLException e)
311         {
```

```
312                     System.out.println(e.getMessage());
313             }
314
315             resTurn.add(aux);
316             resTurn.add(rests);
317             return resTurn;
318      }
319
320     /*****************************************************
321      * Obtiene restaurantes que estan en torno a una direccion
            dada
322      *
323      * @param double x coordenada x (latitud)
324      * @param double coordenada y (longitud)
325      * @return Restaurantes
326      *
327      *****************************************************/
328     private static Restaurantes restaurantesEntornoADir(double
           x, double y)
329     {
330             double margen = 0.003;
331             double xA, xB, yA, yB;
332             String query = "";
333             ResultSet resultSetRest = null;
334             Restaurantes restaurantes = new Restaurantes();
335
336             try
337             {
338                     conectarBBDD();
339
340                     xA = x + margen;
341                     xB = x - margen;
342                     yA = y + margen;
343                     yB = y - margen;
344
345                     Statement statement = cn.createStatement();
346
347                     query = "SELECT * FROM Restaurante WHERE
                           coord_X BETWEEN " + xB + " AND " + xA +"
                            AND coord_Y BETWEEN " + yB + " AND " +
                           yA;
348                     resultSetRest = statement.executeQuery(
                           query);
```

```
349
350                 while (resultSetRest.next())
351                 {
352                         Restaurante rest = new Restaurante(
                               Integer.parseInt(resultSetRest.
                               getString("id")), resultSetRest.
                               getString("nombre"),
                               resultSetRest.getString("
                               direccion"), Double.parseDouble(
                               resultSetRest.getString("coord_X
                               ")), Double.parseDouble(
                               resultSetRest.getString("coord_Y
                               ")), Float.parseFloat(
                               resultSetRest.getString("
                               puntuacion")));
353                         restaurantes.add(rest);
354                 }
355         }
356         catch (ClassNotFoundException CNFE)
357         {
358                 System.out.println(CNFE.getMessage()+"1");
359         }
360         catch (SQLException SQLE)
361         {
362                 System.out.println(SQLE.getMessage()+"2");
363         }
364         catch (InstantiationException IE)
365         {
366                 System.out.println(IE.getMessage()+"3");
367         }
368         catch (IllegalAccessException IAE)
369         {
370                 System.out.println(IAE.getMessage()+"4");
371         }
372         return restaurantes;
373  }
374
375 /**************************************************
376  * Inserta un restaurante en la bbdd
377  *
378  * @param String nombre nombre del restaurante
379  * @param String dir direccion del restaurante
380  * @param double x coordenada x (latitud)
```

```
381   * @param double coordenada y (longitud)
382   * @param String web web del restaurante
383   * @return String json con la respuesta
384   *
385   ************************************************/
386   public static String insertRestaurante(String nombre,
          String dir, double x, double y, String web)
387   {
388           String query = "";
389           String resJson = "{\"update\":\"nok\"}";
390           Statement statement;
391           try
392           {
393                   conectarBBDD();
394                   statement = cn.createStatement();
395
396                   query = "SELECT id FROM Restaurante WHERE
                          nombre = '"+nombre+"' AND " + "direccion
                           = '"+dir+"'";
397                   ResultSet resultSet = statement.
                          executeQuery(query);
398
399                   if(resultSet.next())
400                   {
401                           resJson = "{\"update\":\"nok\"}";
402                   }
403
404                   else
405                   {
406                           query = "INSERT INTO Restaurante (
                                  nombre, direccion, coord_X,
                                  coord_Y, puntuacion) " + "VALUES
                                   ('" + nombre + "','"+dir+"',"+x
                                  +","+y+",-1)";
407                           int resultado = statement.
                                  executeUpdate(query);
408                           if (resultado==1)
409                           {
410                           resJson = "{\"update\":\"ok\"}";
411                           if (web.length()!=0)
412                           {
413                           int idRest=-1;
```

```
414                             query = "SELECT id FROM Restaurante
                                    WHERE nombre = '"+nombre+"' AND
                                    " + "direccion = '"+dir+"'";
415                             resultSet = statement.executeQuery(
                                   query);
416
417                             while (resultSet.next())
418                             {
419                                     idRest = resultSet.getInt("
                                           id");
420                             }
421
422                             query = "INSERT INTO web (uri,
                                   id_restaurante) " + "VALUES ('"
                                   + web + "',"+idRest+")";
423                             resultado = statement.executeUpdate
                                   (query);
424                             if (resultado==1)
425                             {
426                                     resJson = "{\"update\":\"ok
                                           \",\"web\":\"wok\"}";
427                             }
428                             }
429                             }
430                     }
431
432             }
433             catch (ClassNotFoundException CNFE)
434             {
435                     System.out.println(CNFE.getMessage()+"1");
436             }
437             catch (SQLException SQLE)
438             {
439                     System.out.println(SQLE.getMessage()+"2");
440             }
441             catch (InstantiationException IE)
442             {
443                     System.out.println(IE.getMessage()+"3");
444             }
445             catch (IllegalAccessException IAE)
446             {
447                     System.out.println(IAE.getMessage()+"4");
448             }
```

```
449
450         return resJson;
451  }
452
453 /*****************************************************************
454  * Puntua un plato en la bbdd. En caso de no existir el
        plato, lo crea
455  *
456  * @param String nombre nombre del restaurante
457  * @param String plat plato que se quiere puntuar
458  * @param float punt puntuacion del plato
459  * @return String json con la respuesta
460  *
461  *****************************************************************/
462  public static String insertPlato(String nombre, String
        plat, float punt)
463  {
464         String query = "";
465         String resJson = "{\"update\":\"nok\"}";
466         Statement statement;
467         int id;
468         float puntuacion;
469         int resultado;
470
471         try
472         {
473                 conectarBBDD();
474                 statement = cn.createStatement();
475
476                 query = "SELECT id FROM Restaurante WHERE
                        nombre = '"+nombre+"'";
477                 ResultSet resultSet = statement.
                        executeQuery(query);
478
479                 if(!resultSet.next())
480                 {
481                         resJson = "{\"update\":\"nok0\"}";
482                 }
483                 else
484                 {
```

```
485                     int id_restaurante = resultSet.getInt("id")
                           ;
486
487                     query = "SELECT * FROM Plato WHERE plato =
                           '"+plat+"' AND id_restaurante = "+
                           id_restaurante;
488                     resultSet = statement.executeQuery(query);
489                     if(resultSet.next())
490                     {
491                             puntuacion = resultSet.getFloat("
                                calificacion");
492                             id = resultSet.getInt("id");
493                             if (puntuacion != -1)
494                                     puntuacion = (puntuacion+
                                        punt)/2;
495                             else
496                                     puntuacion = punt;
497
498                             query = "update plato set
                                calificacion = "+puntuacion+"
                                where id = "+id;
499                             resultado = statement.executeUpdate
                                (query);
500                     }
501                     else
502                     {
503                             query = "INSERT INTO plato (plato,
                                calificacion, id_restaurante) "
                                + "VALUES ('" + plat + "'," +
                                punt + ","+id_restaurante+")";
504                             resultado = statement.executeUpdate
                                (query);
505
506                     }
507                     if (resultado == 1)
508                     {
509                     float aux = 0;
510                     int i=0;
511
512                     query = "SELECT calificacion FROM Plato
                           WHERE id_restaurante = "+ id_restaurante
                           ;
513                     resultSet = statement.executeQuery(query);
```

```
514
515                 while (resultSet.next())
516                 {
517                         aux = aux + resultSet.getFloat("
                               calificacion");
518                         i = i+1;
519                 }
520                 aux = aux/i;
521                 query = "update restaurante set puntuacion
                       = " + aux + " where id = " +
                       id_restaurante;
522                 resultado = statement.executeUpdate(query);
523                 if (resultado == 1)
524                         resJson = "{\"update\":\"ok\"}";
525                 else
526                         resJson = "{\"update\":\"nok1\"}";
527                 }
528                 }
529
530         }
531         catch (ClassNotFoundException CNFE)
532         {
533                 System.out.println(CNFE.getMessage()+"1");
534         }
535         catch (SQLException SQLE)
536         {
537                 System.out.println(SQLE.getMessage()+"2");
538         }
539         catch (InstantiationException IE)
540         {
541                 System.out.println(IE.getMessage()+"3");
542         }
543         catch (IllegalAccessException IAE)
544         {
545                 System.out.println(IAE.getMessage()+"4");
546         }
547
548         return resJson;
549
550 }
551 }
```

B.2. Código HTML - JSP

index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" %>
<%@ page import = "core.Gestor" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
        <head>
                <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
                <title>resto JSP</title>
        </head>
        <body>
                Proyecto Final de Carrera.
                Fecha Actual:
                <%= new java.util.Date() %>


                <FORM NAME="form1" ACTION="index.jsp" METHOD="get">
                      opcion: <input type="text" name="option">
                      x: <input type="text" name="X">
                      y: <input type="text" name="Y">
                      comida: <input type="text" name="comida">
                      <input type="hidden" name="OK">
                      <input type="submit" value="show">

                </FORM>
<%
if(request.getParameter("OK") != null)
{
        double x;
        double y;
        String comida;
        String nombre;
        String dir;
```

```
33          String web;
34          float puntuacion;
35
36          char opcion = request.getParameter("option").charAt
               (0);
37          switch(opcion)
38          {
39                  case '0':
40                          x = Double.valueOf(request.
                               getParameter("X"));
41                          y = Double.valueOf(request.
                               getParameter("Y"));
42                          comida = request.getParameter("
                               comida");
43                          out.println(Gestor.
                               obtenerRestaurantes(x, y, comida
                               ));
44                  break;
45                  case '1':
46                          nombre = request.getParameter("
                               nombre");
47                          dir = request.getParameter("dir");
48                          x = Double.valueOf(request.
                               getParameter("X"));
49                          y = Double.valueOf(request.
                               getParameter("Y"));
50                          web = request.getParameter("web");
51                          out.println(Gestor.
                               insertRestaurante(nombre, dir, x
                               , y, web));
52                  break;
53                  case '2':
54                          nombre = request.getParameter("
                               nombre");
55                          comida = request.getParameter("
                               plato");
56                          puntuacion = Float.valueOf(request.
                               getParameter("puntuacion"));
57                          out.println(Gestor.insertPlato(
                               nombre, comida, puntuacion));
58                  break;
59                  default:
60                  out.println("No hay texto para eso");
```

```
61              };
62
63      }
64      %>
65
66              </body>
67      </html>
```